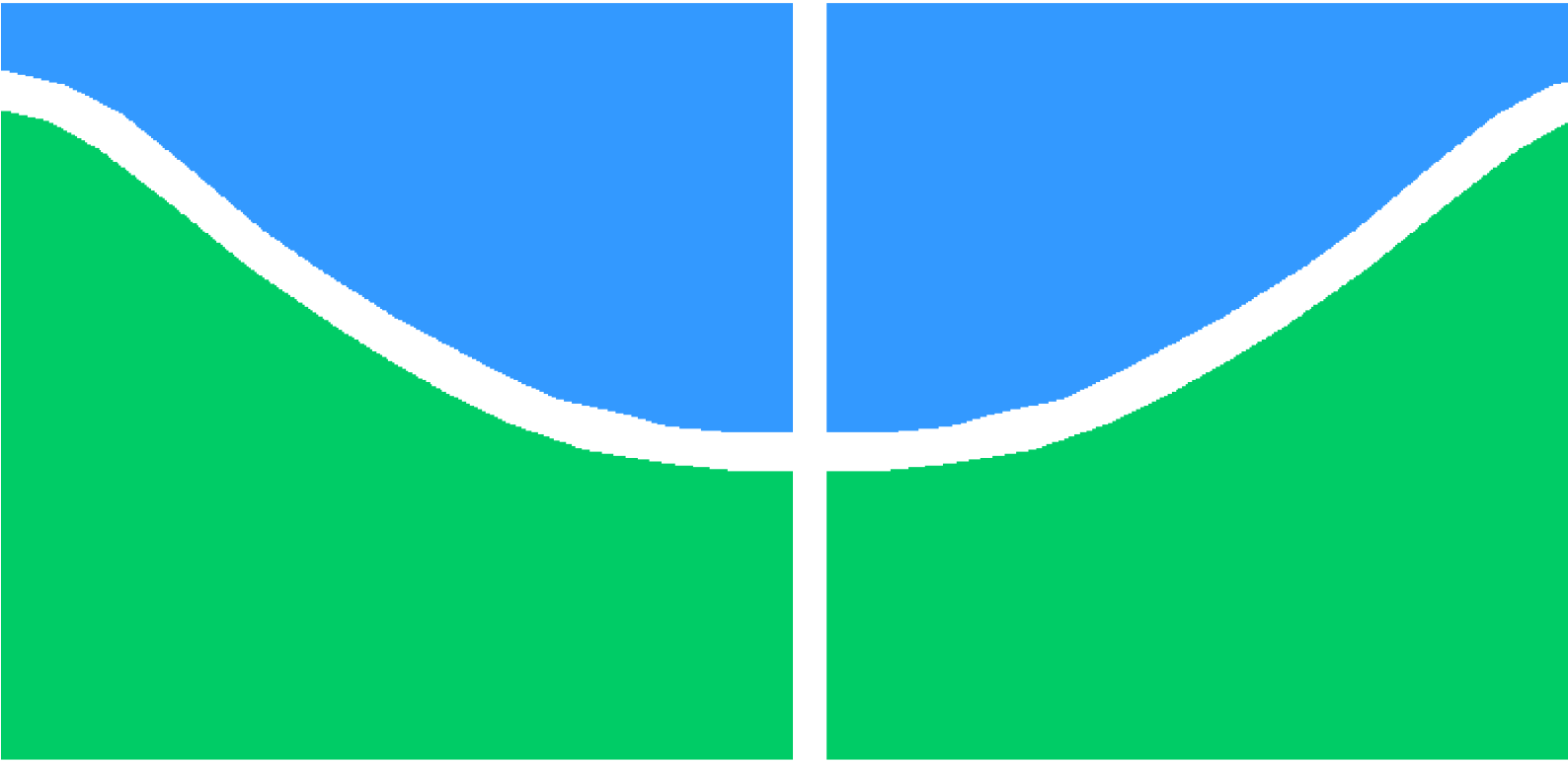


PROJETO DE GRADUAÇÃO

Deep Learning aplicado à previsão de incrustações em condensadores

Por

Matheus Maurício Rodrigues Pereira

Brasília, 27 de maio de 2021

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

PROJETO DE GRADUAÇÃO 2

Deep Learning aplicado à previsão de incrustações em condensadores

Por

Matheus Maurício Rodrigues Pereira

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro Mecânico

Banca Examinadora

Prof. Dr. João Manoel Dias Pimenta, UnB/ENM _____

Prof. Dr. Edgar Amaral Silveira, UnB/ENM _____

Prof. Dr. Mário Benjamin Baptista de Siqueira, UnB/ENM _____

Brasília, 27 de maio de 2021

Resumo

Este trabalho consiste na elaboração de um modelo de *deep learning* capaz de prever a progressão de incrustações em *chillers* resfriados à água, uma vez que atualmente a maior parte das instalações praticamente não possui metodologia para a programação da limpeza dos equipamentos. Como o problema em questão se trata da previsão de uma série temporal, foi realizada a programação e o treino de redes neurais recorrentes do tipo LSTM Bidirecional. O treino das redes foi feito com o uso de dados gerados a partir de um modelo matemático de um *chiller* virtual que considera incrustações de CaCO3 e suas consequências, como a diminuição do diâmetro interno e a perda de carga. Desse modo, a contribuição feita para a evolução do *status quaestionis* se deu por meio da exploração do uso de redes neurais treinadas com dados sintéticos e da preparação do caminho para trabalhos futuros, visando o avanço da aplicação de inteligência artificial ao problema das incrustações.

Palavras-chave: incrustação, trocadores de calor, *chillers*, EES, *machine learning*, LSTM bidirecional.

Abstract

This project consists on the elaboration of a deep learning model capable of predicting the fouling progress on water cooled chillers, as most installations have almost no method for scheduling the equipment cleaning. Since the problem at hand is about predicting a temporal series, it was made the programming and training of recurrent neural networks of the type Bidirectional LSTM. The training of the neural networks was made with data generated from a mathematical model of a virtual chiller that takes into account the progression of CaCO3 fouling and its consequences on the equipment, such as the reduction of the internal diameter and the pressure drop. Therefore, the contribution made to the development of the *status quaestionis* was done by the exploration of neural networks trained with synthetic data and the preparation of the way to future works, aiming at the advance in the application of artificial intelligence to the fouling problem.

Key-words: Fouling, heat exchangers, chillers, EES, machine learning, bidirectional LSTM.

Lista de Figuras

Lista de símbolos

Símbolos Latinos

A	Área
c	Calor específico
c_r	Fator de remoção
D	Diâmetro do tubo
f	Fator de atrito
h	Entalpia
k_f	Condutividade do material incrustante
k_{sp}	Coeficiente de solubilidade
k_R	Coeficiente de precipitação
k_D	Coeficiente do escoamento
$\dot{m}$	Vazão mássica
N_t	Número de tubos
h_t	Estado interno
P	Pressão
Q	Vazão
$\dot{Q}$	Fluxo de calor
Re	Número de Reynolds
R_f	Resistência térmica da camada de incrustação
R_{f_∞}	Valor assintótico de resistência da incrustação
R_g	Constante universal dos gases

R_{tot}	Resistência térmica total
s	Entropia
Sc	Número de Schmidt
T	Temperatura
T_r	Temperatura de referência
t	Tempo
U	Coeficiente global de transferência de calor
u_m	Velocidade média do escoamento
US$	Dólares americanos
v	Volume
$\dot{W}_c$	Potência do compressor

Símbolos Gregos

β	Constante temporal
ε	Efetividade
δ_f	Espessura da camada de incrustação
ΔP	Perda de carga
π	Número pi
ρ	Densidade
ϕ_D	Taxa de deposição
ϕ_R	Taxa de remoção
ψ	Fator de progressão temporal

Subscritos

1	Ponto 1
2	Ponto 2
3	Ponto 3
4	Ponto 4

a	Água
c	Limpo
cd	Condensador
e	Entrada
ev	Evaporador
F	Frio
f	Incrustado
o	Externa
Q	Quente
s	Saída
tot	Total
i	Interna

Siglas e abreviaturas

API	*Application Programming Interface*
AVAC	Aquecimento, Ventilação e Ar Condicionado
CFD	*Computational Fluid Dynamics*
COP	Coeficiente de *Performance*
EES	*Engineering Equations Solver*
PMOC	Plano de Manutenção e Controle
GPU	Unidade de Processamento Gráfico
GRU	*Gated Recurrent Unit*
LMTD	Média Logarítmica das Diferenças de Temperaturas
LSTM	*Long Short Term Memory*
MSE	*Mean Squared Error*
NARX	*Nonlinear Auto Regressive with eXogenous input*
NUT	Número de Unidades de Transferência
ReLU	*Rectified Linear Unit*

RNN	*Recurrent Neural Network*
SGD	*Stochastic Gradient Descent*
TBU	Temperatura de Bulbo Úmido
TEMA	*Tubular Exchanger Manufacturers Association*

Sumário

1 Introdução

1.1 O tema em estudo e sua relevância

O termo incrustação designa o fenômeno da constituição de camadas ou crostas devido a certas substâncias presentes em quantidades excessivas na água, que vão se depositando ou aderindo às paredes dos tubos, especialmente os tubos metálicos, diminuindo o diâmetro interno (NETTO, 2011). Trata-se de um problema silencioso, que passa despercebido para os mais incautos e que tem o sério risco de ser subestimado.

No dia 19 de janeiro de 2004, em uma planta de gás natural liquefeito na Argélia, houve a explosão de uma caldeira que causou a morte de 27 pessoas e ferimentos em 74 outras, além de um prejuízo de aproximadamente 800 milhões de dólares. As investigações indicaram que as incrustações consequentes de uma purga mal executada dificultou a troca de calor nos tubos da caldeira, e uma das hipóteses é que essa camada tenha se soltado e elevado tanto a temperatura quanto a pressão da água além dos níveis esperados, causando uma reação em cadeia que levou à explosão. Apesar desse evento não ser especificamente sobre os tipos de equipamentos estudados neste trabalho, ilustra como as adversidades não vistas podem ser completamente ignoradas, mesmo em casos de alto risco.

Alguns efeitos deste fenômeno podem ser observados no dia a dia de pessoas comuns. A chamada “água dura”, que deixa as torneiras como na Fig. 1, tem esse nome por ter alta concentração de minerais, que formam incrustações nos encanamentos, nas torneiras e pias.

Figura 1 – Torneira com incrustações e oxidação (CANVA, 2020).

Os *chillers* resfriados à água, foco deste trabalho, estão sujeitos ao mesmo problema. Como a água que passa pelo condensador é resfriada na torre de resfriamento, que fica em contato com o ar externo, ocorre a mistura de outros elementos indesejados, tais como bactérias, poeira ou outras partículas. Por isso, faz-se necessária a limpeza dos tubos no interior do equipamento.

Figura 2 – Limpeza de um condensador (CANVA, 2020).

As consequências diretas de se ter tubulações com esse tipo de problema são o aumento da perda de carga do escoamento e o aumento da resistência térmica do sistema.

Por esses motivos, o consumo de energia do sistema aumenta com o passar do tempo, pois os Coeficientes de *Performance*, os COPs, da bomba e do compressor tendem a reduzir continuamente. Ademais, no caso de uma planta industrial, se a manutenção preventiva não for realizada, uma parada não planejada certamente prejudicará toda a produção. Podem acontecer também falhas inesperadas nos equipamentos ou haver a necessidade da substituição de toda a rede de tubos. Resumindo, há um impacto relevante no aumento dos custos de produção e queda da produtividade como um todo. Todos esses gastos com logística, mão de obra e equipamentos acabam refletindo no preço do produto ou serviço, afetando, também, o consumidor final.

Segundo a *Water Conservation Technology International*, combinando-se os dados do *Handbook Carrier* e do livro *Operations and Maintenance Manual for Energy Management* por James Piper, calcula-se que para um *chiller* de $2.000ton$ operando 3.000 horas por ano, a uma avaliação de eficiência de energia de $0,65kW/ton$ e a um custo de US\$$0,09/kW-hr$, os custos de um depósito de apenas $0,006$ polegadas resultam em uma perda de US\$$14.000,00$ por ano. Com uma acumulação de $0,036$ polegadas, o custo da energia utilizada a mais vai para US\$$95.000,00$ por ano.

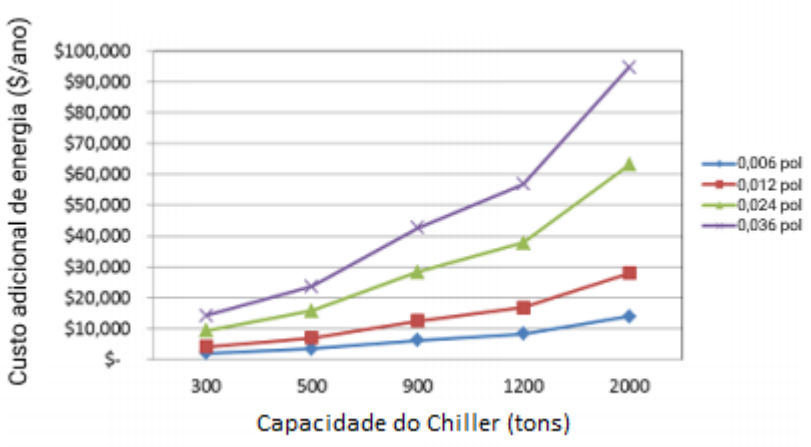


Figura 3 – Custo adicional de energia anual do *chiller* devido a incrustações do condensador (INTERNATIONAL, 2010).

Atualmente existem incontáveis tipos, modelos e séries de máquinas resfriadas à água fria – e novas linhas são lançadas constantemente, por diversas empresas mundo afora. Cada equipamento tem suas várias particularidades, o que torna uma padronização completa praticamente impossível. Além disso, cada um é utilizado em um sítio com diferentes condições climáticas e materiais de qualidades diversas.

Por este motivo, não surpreende que alguns problemas de instalação, operação e manutenção sejam resolvidos com base puramente empírica ou semi empírica, como é o caso das incrustações. Em outras palavras, geralmente sabe-se que os equipamentos precisam ser abertos para a manutenção, mas não se sabe exatamente quando.

A temperatura de *approach*, comumente utilizada por empresas do ramo, fornece

alguma confiabilidade ao processo, mas existem poucas informações para, quer ratificá-la, quer retificá-la.

1.2 Objetivos

O objetivo do presente trabalho é criar um modelo de rede neural artificial capaz de prever a propagação da incrustação. A este objetivo, associam-se, ainda, os seguintes objetivos complementares e metas:

- Desenvolver um modelo matemático para um *chiller* com progressão de incrustação;
- Criar uma estratégia que possa ser usada para a programação de redes neurais em trabalhos futuros;
- Explorar o uso de dados sintéticos para o treino de redes neurais.

Com isso, espera-se contribuir para o avanço no conhecimento da aplicação de inteligência artificial ao problema das incrustações.

1.3 Revisão Bibliográfica

Para a obtenção de informações com fim de projeto e análise de trocadores de calor sujeitos a incrustações, é possível pensar em uma metodologia baseada na experiência prática, como as recomendações da *Tubular Exchanger Manufacturers Association* – TEMA – reportadas por Chenoweth (1988). Na falta de dados reais, essas recomendações oferecem um bom direcionamento para o projeto; porém, em alguns casos, as condições nas quais elas seriam aplicáveis não são explicitadas, e mesmo quando algumas o são, costumam não ser completas. Além disso, não refletem com fidelidade suficiente as condições reais, devido às variações nas propriedades físico-químicas das aplicações, especialmente na quantidade de partículas externas, de microorganismos e composição da água.

Existem também métodos de laboratório. A ideia, então, é simular os fenômenos em um ambiente controlado para que se possa entender quais variáveis afetam o comportamento analisado e de que modo isso ocorre. Patel (1981) sugeriu que fossem utilizadas seções quadradas ou retangulares de escoamento, para se que possa remover uma placa plana para o exame.

O procedimento mais intuitivo de se obter tais informações é medir diretamente a espessura do depósito. Porém, em muitos casos, a espessura do depósito é pequena, talvez menos de 50 micrômetros, então a medição não é fácil de ser obtida. Desse modo, é necessário um micrômetro ou microscópio portátil (BOTT, 1995). Em um caso em que há muita água na composição do depósito, é possível medir a condutividade elétrica, como

descrito por Harty e Bott (1981).

Melo e Pinheiro (1984) discutiram alguns dos métodos mais comuns para os diferentes tipos de medição que podem ser usados para se estimar a incrustação em aparatos de laboratório. Praticamente qualquer medição, sendo ela adequada, pode ser usada para essa estimativa. A pesagem, a medição da espessura, a medição da troca de calor e a medição da queda de pressão são as mais comuns.

Como as incrustações desenvolvem-se lentamente, a medição experimental também é dificultada. Para superar este problema, Fryer e Slater (1987) fizeram uso de uma tubulação cônica, significando em uma variação da velocidade do fluido, bem como da tensão de cisalhamento. O afunilamento significa que a área do fluxo continuamente varia ao longo dos tubos. Isto foi feito para identificar o cisalhamento crítico necessário para manter uma superfície limpa nas condições dadas.

Mott e Bott (1991) realizaram um estudo a partir da medição da massa total do sistema para calcular a deposição de biofilme. Isso é possível ser feito desde que se saiba a densidade do depósito e as dimensões da tubulação, e que preferivelmente sejam usados tubos de paredes finas, para que a massa total seja menor e, portanto, possa ser medida com maior precisão.

Para a medição da transferência de calor, é necessário que se saiba os coeficientes de transferência de calor dos fluidos usados e as temperaturas de entrada e saída de cada um. Realizando-se o balanço de calor entre os dois e considerando o coeficiente global de transferência de calor no sistema incrustado e o seu equivalente no sistema limpo, é obtida uma relação entre esses coeficientes e as espessuras das incrustações, juntamente com suas condutividades térmicas. Observa-se assim que, ao longo do tempo, ocorre a queda do coeficiente global de calor.

Entretanto, uma suposição necessária para esse método é que a presença do depósito não afeta a hidrodinâmica do escoamento. Como resultado, a resistência à incrustação será mais baixa do que se o aumento do nível de turbulência fosse incluído. Além disso, fica-se dependente de temperaturas de entrada constantes. É possível ainda que a turbulência desloque a resistência térmica do depósito e valores negativos de resistência térmica serão calculados. Bott e Gudmundsson (1978) já haviam abordado o problema em um estudo com água geotérmica.

Outro método relevante é a análise da mudança da queda de pressão causada pelo depósito. A forma da curva que relaciona a queda de pressão com o tempo, em geral, segue uma forma assintótica, de modo que o tempo para se chegar à resistência assintótica possa ser determinado. O método é geralmente combinado com a mensuração direta da espessura da camada de depósito. Bemrose e Bott (1983) usaram a variação do fator de atrito para estimar a incrustação de particulado e identificaram que este aumenta assintoticamente até um nível de 50% do seu valor inicial, aumentando também a queda de pressão no equipamento.

Existem ainda relatos do uso de métodos de imagem, como rastreadores radioativos, métodos óticos, filmes de alta velocidade, elipsometria e interferometria. As duas últimas usadas foram usadas por Jenkins (1994).

Quanto aos modelos teóricos de progressão de incrustações, existem diversos tipos, discutidos por Kazi (2012). Em geral, é negligenciada a mudança rugosidade da superfície com a formação de depósito, além de normalmente só um tipo de incrustação ser considerado, as mudanças nas propriedades físicas dos fluidos serem desconsideradas e a camada incrustada ser assumida como homogênea. Também são negligenciados o formato dos depósitos e a variação de velocidade do escoamento por mudança de seção transversal. Não se considera, ainda, o efeito de diferentes mecanismos de incrustação simultâneos, o projeto do equipamento, o aumento da área superficial com a deposição, as propriedades do fluxo com partículas desprendidas, a natureza do processo e as flutuações na operação.

Kern e Seaton (1959) desenvolveram um modelo simples para o processo de incrustação, que leva em conta a curva de comportamento assintótico. Nele, existem duas variáveis centrais: o valor assintótico da resistência à incrustação e uma constante relacionada às propriedades do sistema. A maioria dos outros pesquisadores, desde então, apoiaram-se nessa construção para aprimorá-la.

Watkinson e Martinez (1975), baseando-se na equação fundamental do balanço material, assumiram a taxa de deposição como uma relação entre a chamada constante de reação – que se dá conforme a lei de Arrhenius –, as concentrações de sais e a densidade do incrustante.

Quanto à bibliografia a respeito das simulações de *chillers*, Stoecker (1983) apresentou uma metodologia consistente para a modelagem e simulação de sistemas de refrigeração, capaz de incorporar efeitos de condições externas, do ambiente.

Joffily (2007) fez o uso do software *Engineering Equations Solver*, o EES, e com o auxílio de uma bancada de testes, realizou um estudo que visava realizar a avaliação de uma metodologia de testes de caracterização de desempenho de compressores, de ar condicionado e refrigeradores com o *software* citado. Gregory e Sanford (2008), no livro *Heat Transfer*, abordam a aplicação das seguintes ferramentas computacionais para simulação dos trocadores de calor: Maple, MATLAB, FEHT e EES.

Isoppo, Borges e Mantelli (2009), construíram um modelo de trocadores de calor tubulares, no qual recomendavam dividir o feixe de tubos do modelo em sub-feixes, a fim de simplificar os cálculos. Aguiar e Leal (2014) construíram um modelo de ar condicionado de janela baseado na teoria de Stoecker e Jones. Pereira (2016) realizou uma análise numérica e experimental a partir da modelagem completa de um *chiller* para otimização do consumo de energia em edifícios verdes. Meloni e Sousa (2019) propuseram uma metodologia preditiva para otimização da limpeza de trocadores de calor resfriados à água.

Quanto às redes neurais artificiais, é possível dizer que o primeiro passo foi dado

por McCulloch e Pitts (1943), um neurologista e um matemático que estudavam o funcionamento de neurônios. Eles foram os primeiros a modelar uma rede neural simples com circuitos elétricos. Seis anos depois, na primeira publicação do livro de Hebb (1949), foi aprofundado o estudo, indicando que os caminhos aprendidos pelos neurônios são reforçados cada vez que são utilizados. Em 1956, houve a *Dartmouth Summer Research Project on Artificial Intelligence*, um *workshop* no qual foi dado um passo significativo no desenvolvimento da inteligência artificial e de redes neurais, que incentivou a pesquisa na área. Rosenblatt (1958) começou o desenvolvimento do *Perceptron*, baseado no funcionamento dos olhos de moscas. O resultado desta pesquisa foi construído em *hardware* e é tido como a rede neural artificial mais antiga ainda em uso. Já as redes neurais recorrentes, utilizadas para o processamento de dados sequenciais, foram baseadas no trabalho de Rumelhart, Hinton e Williams (1986).

Um dos problemas desse tipo de rede é que, durante a execução do algoritmo de treino (*backpropagation*), o gradiente pode aumentar ou diminuir muito rapidamente devido à multiplicação sequencial de muitos parâmetros. Solucionando este problema, a arquitetura LSTM (*Long Short Term Memory*), ou memória de curto prazo longa, foi proposta por Hochreiter e Schmidhuber (1997), em um trabalho extraordinariamente à frente do tempo. Esse tipo de rede foi aprimorada e é amplamente utilizada ainda hoje. Mais recentemente, este tipo de arquitetura teve seus parâmetros e funcionamento estudados por Greff et al. (2016), mas foi notado que os parâmetros tradicionais funcionam bem, e que é mais importante ajustar de maneira adequada os parâmetros escolhidos arbitrariamente, como número de nós da rede. A arquitetura GRU (*Gated Recurrent Unit*), proposta por Cho et al. (2014), teve seus resultados comparados com os das redes LSTM por Chung et al. (2014) e Jozefowicz, Zaremba e Sutskever (2015), mas não foi percebida uma vantagem clara em se utilizar uma outra em casos gerais, uma vez que ambas arquiteturas fornecem desempenhos comparáveis. Apesar disso, uma diferença importante a ser notada é que redes GRU podem ser treinadas um pouco mais rapidamente e podem precisar de menos dados, mas as do tipo LSTM podem fornecer resultados melhores para maiores bancos de dados.

Quanto ao uso das redes neurais para simulações de fluidos, Tompson et al. (2017) investigaram a possibilidade de realização de simulações de *Computational Fluid Dynamics* – CFD – com o auxílio do *machine learning*. Neste trabalho, uma rede neural "assistia" a uma simulação de fluido até certo ponto e continuava-a automaticamente. Os resultados foram surpreendentes e confirmaram a viabilidade do uso desse tipo de algoritmo para o estudo de escoamentos.

A Fig. 4 é uma captura de tela de um vídeo sobre a simulação feita por Gonzalez et al (2020) de um escoamento complexo, mostrando excelente desempenho e fidedignidade. As vantagens desse tipo de procedimento consistem, principalmente, na sua capacidade de generalização, ou seja, na capacidade de reconhecer casos não vistos anteriormente, pois apreende os conceitos gerais e replica-os com facilidade. Para isso, é necessário um

treino extenso, mas que só precisa ser feito uma vez. Isso permite uma velocidade de processamento muito superior à dos sistemas convencionais.

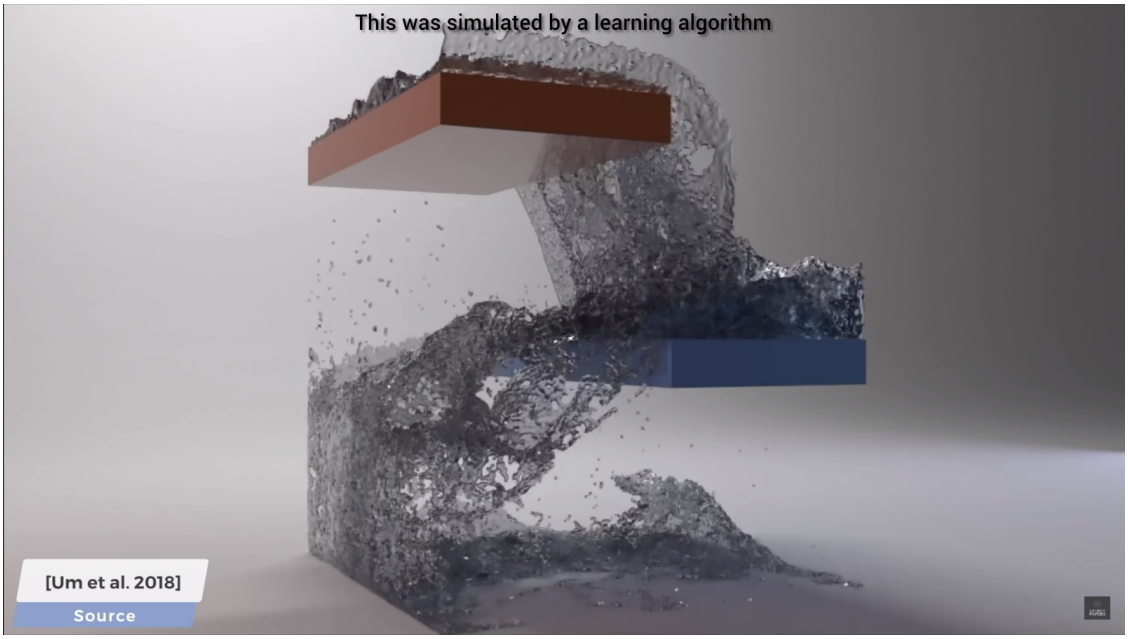
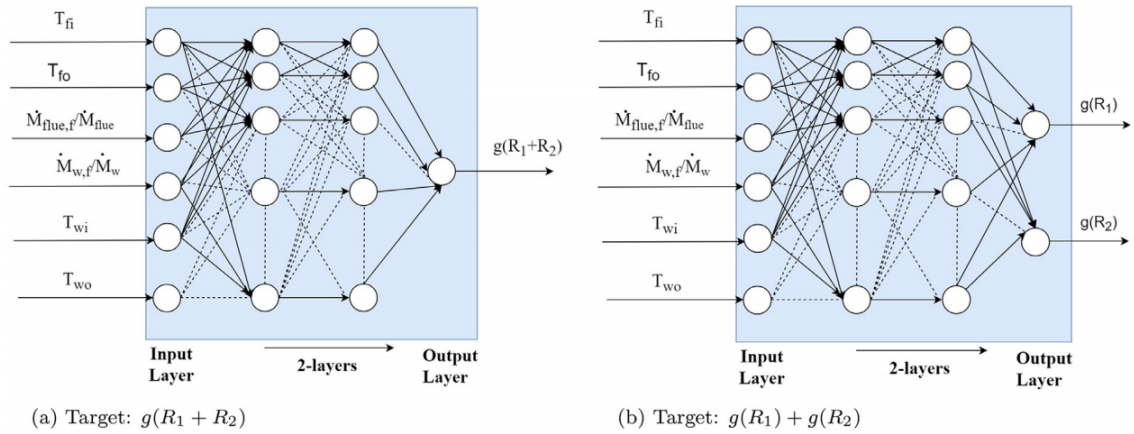


Figura 4 – Exemplo de simulação de um escoamento complexo por inteligência artificial (TWOMINUTEPAPERS, 2020).

Quanto ao uso das redes neurais para estudo das incrustações, Ibrahim (2012) construiu um algoritmo que visava predizer a taxa de transferência de calor e fator de incrustação em um trocador de calor do tipo *shell and tube*, que era usado como um dispositivo de pré-aquecimento de nafta.

Biyanto et al. (2015) utilizaram as redes neurais *Multi Layer Perceptron* com o tipo de modelo sendo o *Nonlinear Auto Regressive with eXogenous input* (NARX), além do software HYSYS para analisar o problema em um trem de pré-aquecimento bruto. Sundar et al. (2020) desenvolveram um modelo estatístico escalável e generalizado para predição da resistência à incrustação usando parâmetros comumente medidos de trocadores de calor industriais. Este trabalho oferece um bom direcionamento do caminho a ser seguido: a programação e o treino de uma rede neural capaz de prever, conforme mostrado na Fig. 5, tanto a resistência global à incrustação (esquerda) quanto as individuais de cada fluido (direita), usando como *inputs* as temperaturas de entrada, as temperaturas de saída e as razões dos quocientes de vazão de fluidos no caso incrustado para o caso limpo. A arquitetura foi de uma rede neural artificial de *feedforward* com duas camadas escondidas e que fazia uso da função da ativação ReLU (*Rectified Linear Unit*), ou função linear retificada.



(a) Target: $g(R_1 + R_2)$ (b) Target: $g(R_1) + g(R_2)$

Figura 5 – Exemplo de rede neural para previsão de incrustações (SUNDAR et al., 2020).

Como pode ser visto, o atual estágio de conhecimento em torno do tema central deste trabalho revela uma certa carência de dados confiáveis para o treino das redes neurais, além de testes da conferência da capacidade preditiva destas em instalações reais.

1.4 Estrutura do relatório

No primeiro capítulo, é apresentada a contextualização do problema e de suas discussões, os avanços propostos neste trabalho e uma breve revisão bibliográfica das áreas de incrustações, modelos matemáticos de *chillers* e inteligência artificial. Estes assuntos são aprofundados no capítulo 2, que traz o referencial teórico. No capítulo 3, é primeiramente explicada a abordagem utilizada para que fossem atingidos os objetivos do trabalho, seguida da exposição a respeito da elaboração do modelo matemático e da programação da rede neural. No capítulo 4, são apresentados os resultados obtidos pelo modelo matemático resolvido numericamente e pela rede neural. No capítulo 5, são enunciadas as contribuições aqui obtidas e feitas propostas para trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Unidades resfriadoras de líquido

Uma unidade resfriadora de líquido, normalmente conhecida como *chiller*, é um tipo de equipamento de ar condicionado de grande porte que pode funcionar por absorção ou por compressão de vapor. O grupo dos que funcionam por compressão subdivide-se em duas categorias: os refrigerados à água e refrigerados a ar. Estes rejeitam o calor diretamente para o ambiente externo e geralmente são posicionados na laje das edificações, enquanto aqueles necessitam de torres de resfriamento para essa função e geralmente são posicionados no subsolo, com bombas que enviam a água para o restante do prédio.

Para que isso seja possível, há o funcionamento de dois ciclos de água; um de água gelada, que circula entre os *fan coils* e o evaporador, e outro que circula entre a torre de resfriamento e o condensador. O primeiro permite que a água resfriada no evaporador seja utilizada para resfriar os ambientes da instalação enquanto o segundo permite que o calor seja rejeitado para o ambiente externo.

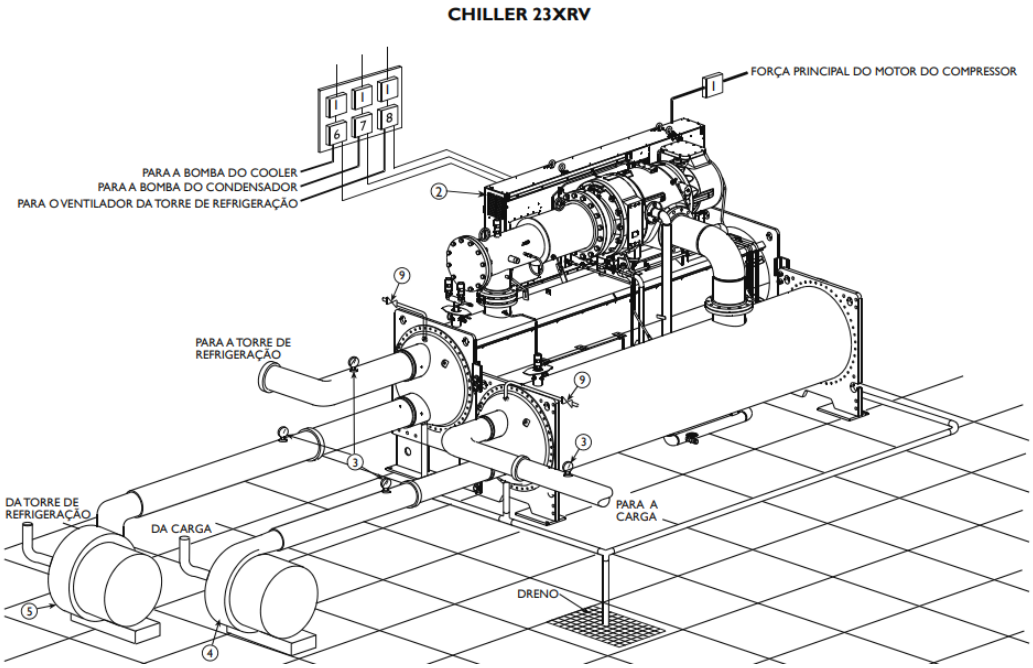


Figura 6 – *Chiller* resfriado à água (CARRIER, 2020).

O funcionamento dos equipamentos resfriados à água acontece conforme esquematizado na Fig. 7.

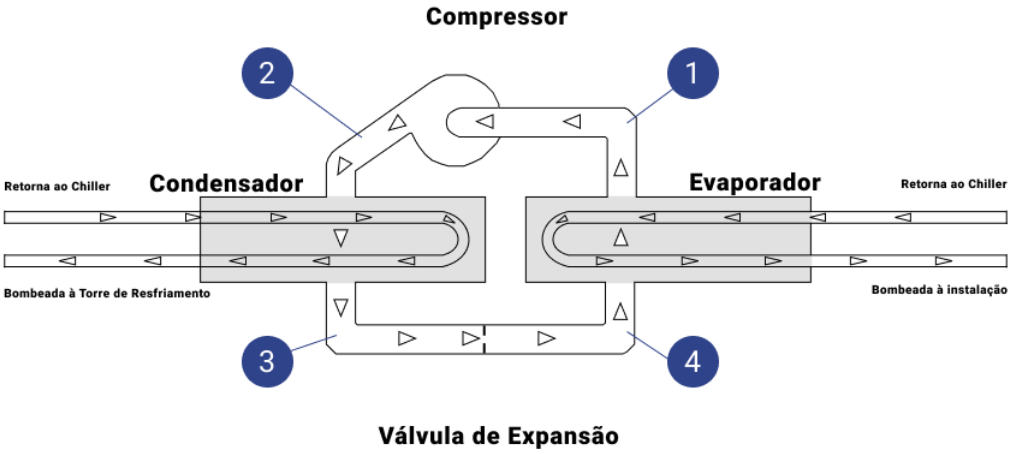


Figura 7 – Esquema do funcionamento de um *chiller*.

Pode-se dizer que o compressor é o coração deste equipamento. Ele faz com que o fluido refrigerante circule entre as quatro etapas do processo: em direção ao condensador, à válvula de expansão, ao evaporador e de volta ao compressor. Durante metade desse processo, o fluido refrigerante está em estado gasoso, e na outra metade, em estado líquido. O condensador é um trocador de calor casco e tubo e, como o nome sugere, tem a função de condensar o fluido que vem do compressor; em seguida, a válvula de expansão expande o refrigerante, que adquire maior capacidade de absorção de calor e segue em direção ao evaporador, que é outro trocador de calor do mesmo tipo e, como o nome também sugere, evapora o líquido que será novamente comprimido.

Para entender melhor como funciona todo esse processo, é necessário que seja levado em conta o ciclo de Carnot. Nele, existem 4 etapas. A primeira é a compressão adiabática e reversível (sem atrito); a segunda é a rejeição de calor a temperatura constante; a terceira é a expansão adiabática e reversível em um motor térmico; e a quarta é a remoção isotérmica de calor de um ambiente a baixa temperatura. Esquematizados na Fig. 8, os processos 1-2 e 3-4 para o ciclo de Carnot ocorrem adiabática e reversivelmente, ou seja, a entropia permanece constante.

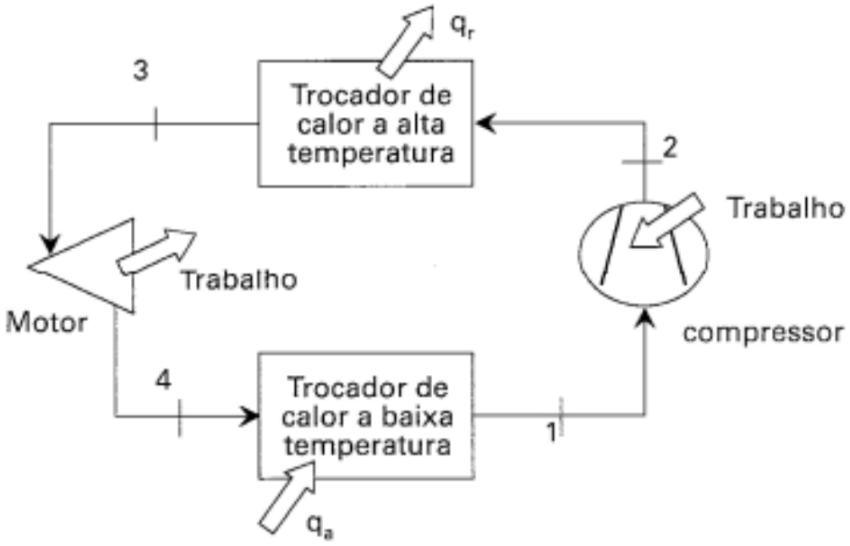


Figura 8 – O circuito frigorífico de Carnot (STOECKER, 1983).

Entretanto, para um ciclo termodinâmico real, não é possível que a entropia continue constante, pois os processos sempre possuem fatores dissipativos. Além disso, dois processos precisam ser adaptados: a expansão e a compressão. Quanto à compressão, é necessário um cuidado com o fluido para que não haja compressão de vapor úmido (que acarretaria em problemas no equipamento). Por esse motivo, o fluido entra no compressor em estado de vapor superaquecido. Quanto à expansão, substitui-se o motor pela válvula de expansão devido à dificuldade de se desenvolver um motor que trabalhe com um fluido líquido-vapor, à dificuldade de controlá-lo e à dificuldade de integrá-lo ao compressor. A Fig. 9 mostra um Ciclo Padrão de Compressão a Vapor que contempla as mudanças citadas.

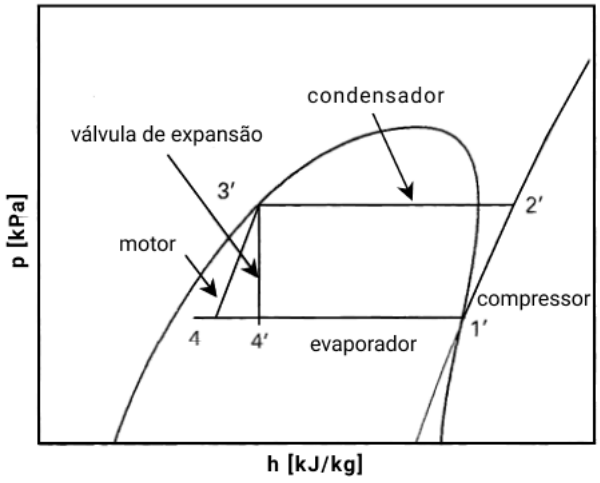


Figura 9 – Ciclo Padrão de Compressão a Vapor (STOECKER, 1983).

Para investigar esse tipo de problema, é necessário que sejam feitos balanços de energia e de massa, analisando suas propriedades termodinâmicas a cada etapa do sistema. A solução mais usual considera um estado de regime permanente no volume de controle. Vale, portanto, a Eq. 2.1.

$$\frac{\partial E_{vc}}{\partial t} = \dot{Q}_{vc} - \dot{W}_{vc} + \sum \dot{m}_e \left(h_e + \frac{v_e^2}{2} + g z_e \right) - \sum \dot{m}_s \left(h_s + \frac{v_s^2}{2} + g z_s \right) \tag{2.1}$$

A eficiência de ciclos é normalmente definida como a relação entre a energia útil, que é o objetivo do ciclo, e a energia que deve ser "paga" para a obtenção do efeito desejado. Assim, o COP é dado pela Eq. 2.2. Nela, $\dot{W}_c$ é a potência do compressor e $\dot{Q}$ o fluxo de calor.

$$COP = \frac{\dot{Q}}{\dot{W}_c} \tag{2.2}$$

Esta equação pode também ser reescrita na forma da Eq. 2.3, na qual as variáveis h de 1 a 4 representam a entalpia de cada estado.

$$COP = \frac{h_1 - h_4}{h_3 - h_1} \tag{2.3}$$

Quanto à manutenção dos *chillers*, a forma mais comum de planejamento para os equipamentos de água gelada incrustados é a temperatura de *approach*. É comum que seja recomendado a monitoramento desta temperatura e que se faça a limpeza quando ela chega a um valor previamente determinado.

No evaporador, o *approach* indica a diferença entre a temperatura de saída de água gelada e a temperatura de evaporação do fluido refrigerante; enquanto isso, no condensador, indica a diferença entre a temperatura de condensação do fluido refrigerante e a temperatura de saída de água de resfriamento. Quanto menor for o *approach* de projeto no condensador, menor a potência absorvida no motor elétrico, maior o COP do *chiller*, menor a temperatura de condensação e menor o *lift* do compressor – o que também demonstra o impacto dos depósitos na tubulação.

Existe, também, o Plano de Manutenção e Controle - PMOC, que é obrigatório por lei federal (BRASIL, 2018) e define as diretrizes de manutenção e controle de sistemas de ar condicionado. A recomendação usual é que o condensador seja limpo anualmente.

Entretanto, essa limpeza pode ser necessária com maior frequência, caso a incrustação seja significativa nesse período.

2.2 Trocadores de calor

Trocadores de calor são usualmente classificados em função da configuração do escoamento e do tipo de construção. Os mais comuns envolvem escoamentos no mesmo sentido, em sentidos opostos ou em escoamento cruzado. A configuração pode ser com ou sem aletas e, no caso de equipamentos tubulares, pode haver um ou mais passes. A Fig. 10 mostra um trocador de calor do tipo casco e tubo com um passe tanto no casco quanto nos tubos.

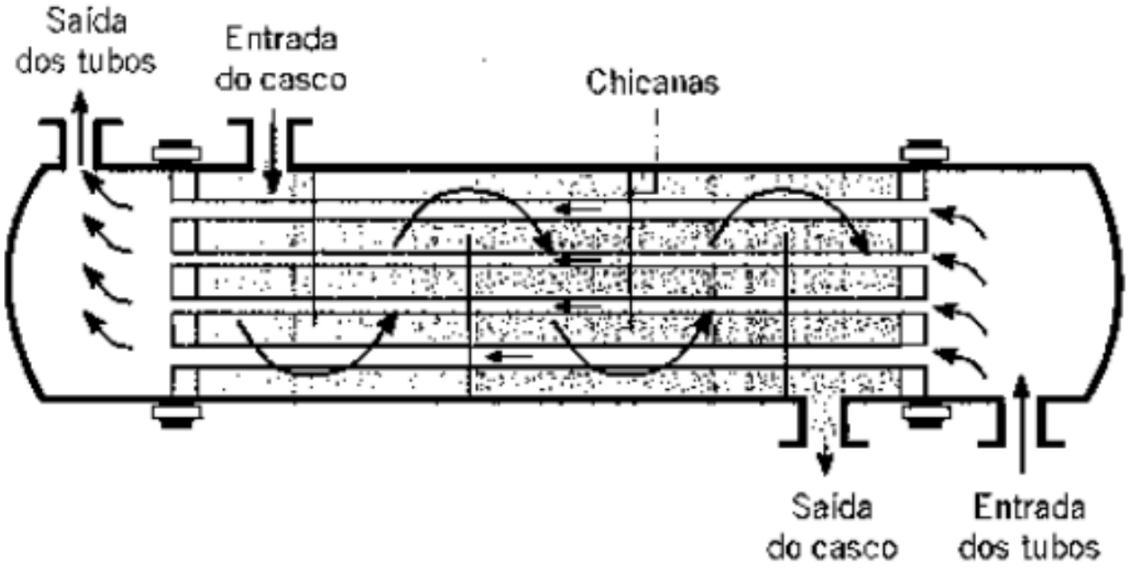


Figura 10 – Trocador de calor casco e tubos com um passe no casco e um passe nos tubos (BERGMAN et al., 2008).

Independente da configuração do tipo ou geometria, um balanço de energia pode ser escrito para um volume de controle que permeia o trocador de calor. A energia cinética e potencial dos fluxos geralmente são negligenciáveis e é assumido um regime permanente. Gregory e Sanford (2008) mostram a análise a partir do balanço de energia, que, neste caso, simplifica-se a:

$$\dot{m}_Q h_{Q,e} + \dot{m}_F h_{F,e} - \dot{m}_Q h_{Q,s} - \dot{m}_F h_{F,s} + \dot{Q}_{viz} = 0 \tag{2.4}$$

Na qual $\dot{m}$ é a vazão mássica, h é a entalpia e $\dot{Q}_{viz}$ é o fluxo de calor perdido para a vizinhança. Os subscritos "Q" e "F" significam, respectivamente, "Quente" e "Frio"; enquanto os "e" e "s", "Entrada" e "Saída" do trocador. Se a capacidade de calor específico c é constante, então a entalpia do fluido pode ser escrita como um produto entre calor específico à pressão constante e a temperatura T relativa a uma temperatura arbitrária de referência T_r.

$$h = c(T - T_r) \tag{2.5}$$

Em muitos casos, a dependência da capacidade de calor específico em relação à temperatura é pequena. É assumido, também, que a taxa de transferência de calor à vizinhança é pequena o suficiente para ser desconsiderada. Deste modo:

$$\dot{Q} = \dot{m}_Q c_Q (T_{Q,e} + T_{Q,s}) = \dot{m}_F c_{F,s} (T_{F,s} - T_{F,e}) \tag{2.6}$$

O desempenho do trocador de calor pode apenas ser predito pela derivação e resolução da equação diferencial governante que concerne à transferência de calor local entre os fluidos e a temperatura associada de cada escoamento. A solução para essa equação fornece uma segunda relação independente entre as temperaturas de entrada e saída dos fluidos quente e frio. Felizmente, as soluções para as equações governantes que são associadas a configurações de troca de calor que são comumente encontradas já foram obtidas. Essas soluções estão disponíveis de duas formas: a Média Logarítmica das Diferenças de Temperaturas – LMTD – e a efetividade-NUT (GREGORY; SANFORD, 2008).

O método LMTD expressa a taxa de transferência de calor entre os dois escoamentos em um trocador de calor como o produto da diferença de temperatura pelo produto UA do trocador, chamado de condutância. A grandeza U é definida como o coeficiente global de transferência de calor enquanto A é a área superficial em que ocorre a troca de calor. A expressão utilizada é análoga à lei do resfriamento de Newton:

$$\dot{Q} = UA\Delta T_{LMTD} \tag{2.7}$$

Para os trocadores de calor de casco e tubo, a solução analítica desta equação é feita considerando que a LMTD é corrigida por um fator F, conforme a Eq. 2.8, uma vez que nesses casos ela sempre será menor que nos casos de escoamentos contrário e paralelo, que seriam usados no cálculo padrão. A grandeza corrigida é chamada $\Delta T_{LMTD,c}$.

$$\Delta T_{LMTD,c} = F\Delta T_{LMTD} \tag{2.8}$$

No que se refere ao método da efetividade NUT, segundo Bergman et al. (2008), a efetividade para um condensador de *chiller*, será conforme as Eq. 2.9 e 2.10.

$$\varepsilon = 1 - \exp(-NTU) \tag{2.9}$$

$$NTU = \frac{UA}{C_{min}} \tag{2.10}$$

E a troca de calor será dada por:

$$\dot{Q} = \varepsilon C_{min} (T_{Q,e} - T_{F,e}) \tag{2.11}$$

A Fig. 11 ilustra um caso genérico no qual existe mais do que uma parede entre os dois fluidos e, por esse ponto de vista, a resistência térmica comporta-se de modo semelhante à resistência elétrica. Quando existem resistores elétricos associados em série, as resistências de todos eles devem ser somadas para que seja obtida a resistência total. De modo análogo, as resistências térmicas de paredes compostas são somadas. Isto é possível também para contabilização da deposição, uma vez que aos poucos forma-se uma nova parede no interior dos tubos.

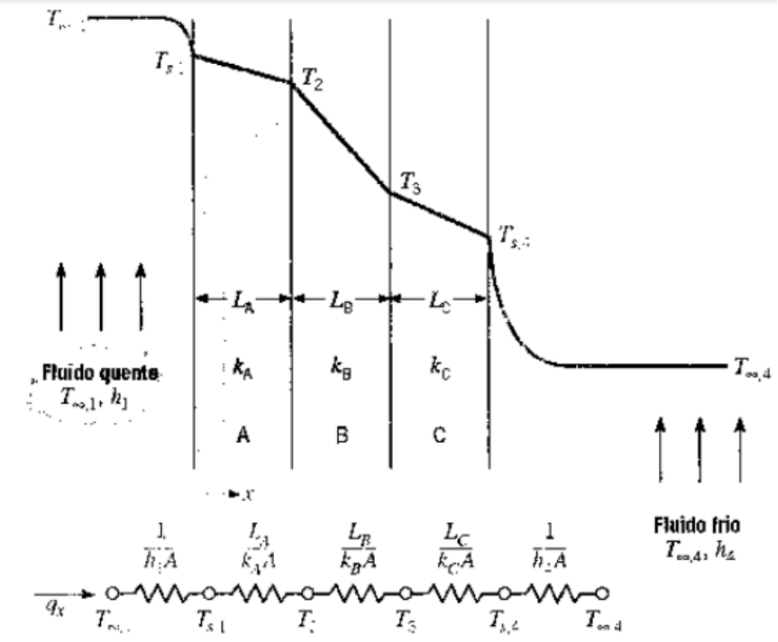


Figura 11 – Circuito térmico equivalente para uma parede composta em série (BERGMAN et al., 2008).

Partindo deste princípio, o coeficiente global de transferência de calor pode, no caso de trocadores de calor de casco e tubo sem aletas, ser calculada conforme as Eq. 2.12, Eq. 2.13, Eq. 2.14 e Eq. 2.15. Nelas, R_{tot} é a resistência térmica total, D é o diâmetro, A é a área, k é a condutividade térmica do material do trocador de calor, L é o comprimento total da tubulação, h_t é o coeficiente de transferência de calor por convecção e os subscritos "i" e "o" referem-se às superfícies internas e externas.

$$R_{tot} = \frac{\Delta T}{\dot{Q}} = \frac{1}{UA} \tag{2.12}$$

$$\frac{1}{UA} = \frac{1}{h_{t,i}A_i} + \frac{R_f}{A_i} + \frac{\ln \frac{D_o}{D_i}}{2\pi kL} + \frac{1}{h_{t,o}A_o} \tag{2.13}$$

$$A_i = \pi D_i L \tag{2.14}$$

$$A_i = \pi D_o L \tag{2.15}$$

No caso do *chiller* resfriado à água, existem dois trocadores de calor casco e tubo: o condensador e o evaporador. A troca de calor no condensador ocorre como ilustrado pela Fig. 12. Nela, o interior do tubo foi ilustrado com um gradiente de coloração para que seja representado o gradiente de temperatura da água, que absorve gradualmente o calor do refrigerante e em seguida tem sua temperatura diminuída na torre de resfriamento. Como esta água entra em contato com o ar externo, a tubulação no interior do trocador de calor está sujeita às incrustações.

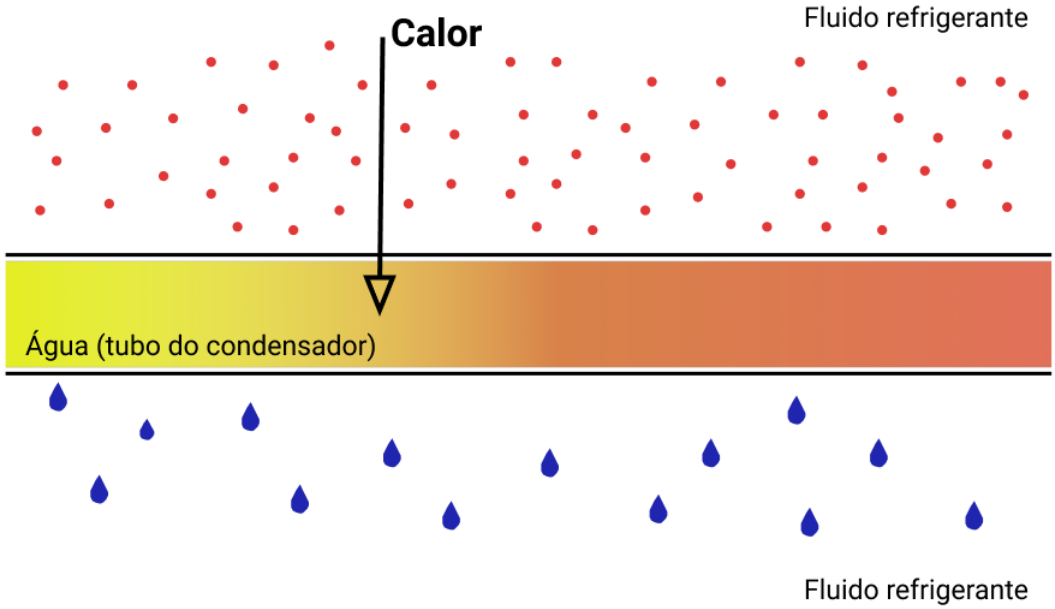


Figura 12 – Troca de calor no condensador.

2.3 Progressão de incrustações

Existem diversos tipos de incrustações possíveis. Segundo Awad (2011), a incrustação de particulado é o tipo de deposição de partículas suspensas no fluido em contato com as superfícies do trocador de calor. Caso a deposição ocorra também devido à gravidade, o resultado é chamado de sedimentação. É influenciada pelos seguintes fatores: concentração de partículas suspensas, velocidade do escoamento, condições de temperatura na superfície (se aquecido ou não) e fluxo de calor na superfície.

A cristalização ou precipitação ocorre por sais dissolvidos em soluções saturadas devido a mudanças de solubilidade com a temperatura e, consequentemente, a precipitação na superfície do trocador de calor. A deposição de sais inversamente solúveis na superfície aquecida origina uma camada dura e firme, enquanto em superfícies resfriadas, geralmente tem poros e camadas semelhantes à lama. É comum quando usada água não tratada, água do mar, água geotérmica, salmoura, soluções aquosas de soda cáustica ou outros sais. É importante ressaltar que o crescimento dos cristais se dá a partir da nucleação.

A incrustação de reações químicas acontece por uma ou mais reações químicas indesejadas entre reagentes contidos no fluido no qual o material não é reagente ou participante – apesar de poder atuar como catalisador, assim como a temperatura. É comumente achada em aplicações da indústria de processos.

A incrustação por corrosão envolve uma reação química ou eletroquímica entre a superfície de troca de calor e o escoamento, produzindo produtos de corrosão. Estes mudam as características térmicas da superfície e a tornam mais propensas à incrustação. É prevalente em aplicações nas quais ocorre reação química e a camada protetora de óxido não se forma.

A incrustação biológica é um fenômeno no qual ocorre deposição de organismos microscópicos ou macroscópicos (normalmente a primeira precede a segunda) e de seus produtos, que se ligam às paredes do trocador de calor. É geralmente chamada "bioincrustação" e costuma ser um problema em fluxos de água na forma de biofilme ou limo, sendo de difícil remoção.

Por último, a incrustação por solidificação consiste no congelamento de um líquido puro ou de componentes de ponto de fusão mais alto em uma solução de múltiplos componentes em superfícies sub-resfriadas. Este processo ocorre em temperaturas baixas, normalmente em temperatura ambiente ou abaixo, a depender da pressão do local. Os principais fatores que o afetam são o fluxo de massa do fluido de trabalho, temperatura e condições de cristalização, condições de superfície e concentração do precursor sólido no fluido.

Os principais mecanismos de deposição e remoção estão ilutrados na Fig. 13.

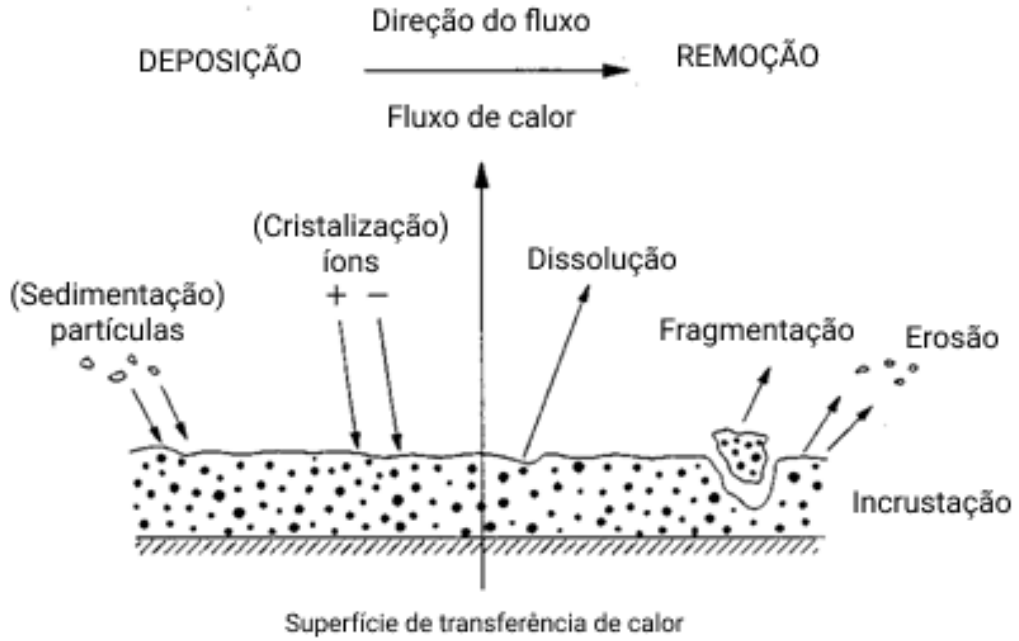


Figura 13 – Deposição e remoção de material incrustado (KAZI, 2012).

Segundo Firdaus, Prasetyo e Luciana (2016), nos trocadores de calor ocorre principalmente a incrustação por cristalização e por deposição de particulado. E segundo Shen et al. (2015), verifica-se que o mecanismo de particulado é o dominante no processo. Com isso, a maioria das pesquisas desenvolvidas aborda o mecanismo particulado.

A primeira grandeza importante para o assunto é a variação de massa do material incrustado, $\frac{dm}{dt}$. Conforme mostrado na Eq. 2.16 e na Fig. 13, ela é simplesmente uma subtração da taxa de deposição ϕ_D pela taxa de remoção ϕ_R.

$$\frac{dm}{dt} = \phi_D - \phi_R \tag{2.16}$$

As consequências mais significativas da acumulação de depósitos são a redução do coeficiente global de transferência de calor (U), a diminuição da efetividade do trocador de calor (NUT), aumento da temperatura de condensação e aumento no consumo de energia da bomba de água devido à perda de carga crescente (menor seção de passagem, que leva à maior velocidade e aumento no fator de atrito pela maior rugosidade). As incrustações aumentam com o tempo e, por esse motivo, é preciso levar em conta o custo de oportunidade de realização a manutenção do *chiller* – uma vez que para isso é necessário parar o equipamento. Deve-se observar, então, se o custo gerado pela incrustação supera o custo gerado pela operação de manutenção.

Segundo Bott (1995), a resistência da transferência térmica à camada de incrustação, R_f, é expressa pela Eq. 2.17, na qual δ_f é a espessura do sólido e k_f a condutividade térmica de um sólido particular.

$$R_f = \frac{\delta_f}{k_f} \tag{2.17}$$

Pode-se calcular essa resistência também a partir do balanço de calor entre os dois fluidos que escoam pelo equipamento, considerando que U_f é o coeficiente de transferência de calor global no sistema incrustado e U_c é o seu equivalente no sistema limpo (e os subscritos 1 e 2 são relativos às camadas internas e externas). Assim, é obtido que:

$$\frac{1}{U_f} - \frac{1}{U_c} = R_f \tag{2.18}$$

Kazi (2012) traz que a perda de pressão em trocadores de calor é considerada mais crítica que a perda na transferência de calor. A camada depositante aumenta a dureza da superfície e diminui o diâmetro interior dos tubos, aumentando a perda de carga. Esta pode ter suas condições limpa e incrustada relacionadas pela Eq. 2.19. Nela, ΔP é a perda de carga, f é o fator de atrito, D é o diâmetro do tubo e u_m é a velocidade média do escoamento.

$$\frac{\Delta P_f}{\Delta P_c} = \frac{f_f}{f_c}\left(\frac{D_c}{D_f}\right)\left(\frac{u_{m,f}}{u_{m,c}}\right)^2 \tag{2.19}$$

Considerando que os fluxos de massa nas condições limpa e incrustada são os mesmos, a equação pode ser reescrita como:

$$\frac{\Delta P_f}{\Delta P_c} = \frac{f_f}{f_c}\left(\frac{D_c}{D_f}\right)^5 \tag{2.20}$$

Além disso, a magnitude de D_f pode ser obtida pela Eq. 2.21.

$$D_f = D_c \exp\left(-\frac{2k_c R_f}{d_c}\right) \tag{2.21}$$

E a espessura δ_f da camada de incrustação pode ser obtida pela Eq. 2.22.

$$\delta_f = 0,5 D_c \left[1 - \exp\left(-\frac{2k_f R_f}{D_c}\right)\right] \tag{2.22}$$

Tratando-se da progressão do fator de incrustação, sabe-se que, idealmente, esta segue uma de três diferentes curvas, conforme mostrado na Fig. 14.

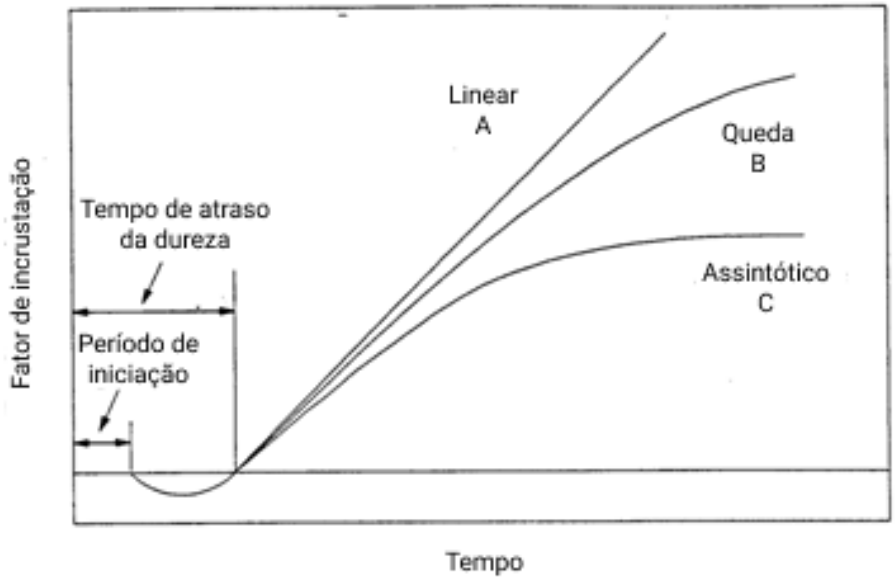


Figura 14 – Evolução do fator de incrustação com o tempo (BOTT, 1995).

A curva C representa a curva assintótica, a curva A representa uma relação linear entre a espessura do depósito com o tempo e a curva B representa uma taxa de deposição em queda após um período linear. É possível que a curva B seja essencialmente parte de uma curva similar à C, e se a continuação do processo fosse permitida, seria observado um valor assintótico.

Kern e Seaton (1959), a partir da equação $\frac{dm}{dt}$ chegaram à Eq. 2.23, que descreve o comportamento da curva assintótica. Nela, R_{f_∞} representa o valor assintótico da resistência causada pela incrustação e β representa uma constante temporal.

$$R_{f_t} = R_{f_\infty}\left(1 - e^{\beta t}\right) \tag{2.23}$$

Os demais modelos desenvolvidos geralmente baseiam-se neste e o desenvolvem. O modelo proposto por Cremaschi e Wu (2015) e usado por Meloni e Sousa (2019), traz para a análise a concentração de minerais na água, com resultados teóricos acurados. Sua base é a Eq. 2.16, que pode ser desenvolvida como:

$$\frac{dR_f}{dt} = \frac{\dot{m}_d - \dot{m}_r}{\rho_f k_f} \tag{2.24}$$

A taxa de deposição, neste caso, é calculada como mostrado na Eq. 2.25. Nela são usadas as concentrações de $Ca^{(2+)}$ e de $CO_3^{(2-)}$. Além disso, k_{sp} é o coeficiente de solubilidade do carbonato de cálcio de valor $10^{(-9)}mol^2/L^2$; k_R é o coeficiente de precipitação (dependente da temperatura) dado pela Eq. 2.26; k_D é um coeficiente que depende do escoamento dado pela Eq. 2.27. Nas duas últimas equações, R_g é a constante universal dos gases, T é a temperatura de mistura da água, Sc é o número de Schmidt e Re é o número de Reynolds.

$$\dot{m}_d = \frac{k_d[CO_3^{(2-)}]\left(1 - \frac{k_{sp}}{[Ca^{(2+)}][CO_3^{(2-)}]}\right)}{1 + \frac{k_p}{k_R[CO_3^{(2-)}]} + \frac{[CO_3^{(2-)}]}{[Ca^{(2+)}]}} \tag{2.25}$$

$$k_R = \exp\left(38,74 - \frac{20700}{R_g T}\right) \tag{2.26}$$

$$k_D = 0,023uRe^{-0,17}Sc^{-0,67} \tag{2.27}$$

Para determinação das concentrações, foi observado que o carbonato de cálcio tem peso molar de $100,0869g/mol$ e que o cátion de cálcio é responsável por $40,04\%$ do peso.

Para que se conheça a taxa de remoção $\dot{m}_r$ do filme, é usada a Eq. 2.28. Nela, c_r é o fator de remoção, que por sua vez é calculado na Eq. 2.29, δ_f é a espessura do depósito e $\rho_f \delta_f^{1,5}$ é a massa instantânea do material depositado. A condutividade do material incrustante k_f foi admitida com o valor de $2,19W/mC$ e $\psi(t)$, o fator de progressão temporal, é dado pela Eq. 2.30 e tem valor mínimo de $0,002$.

$$\dot{m}_r = c_r \rho_f \delta_f^{1,5} \tag{2.28}$$

$$c_r = \frac{0,00212u^2}{k_f^{0,5}\psi} \tag{2.29}$$

$$\psi(t) = 0,99^{(t-1)} \tag{2.30}$$

Desse modo, as equações 2.25 até 2.30 são aplicadas na Eq. 2.24, que deve então ser integrada para a obtenção de R_f em função do tempo.

2.4 Inteligência Artificial

A inteligência artificial é um campo da ciência da computação que se refere a uma inteligência semelhante à humana em computadores, robôs ou outras máquinas. Uma das suas ramificações é o aprendizado de máquina, ou *machine learning*. Segundo Hurwitz e Kirsch (2018), esta é uma forma de inteligência artificial que permite que um sistema aprenda através de dados, ao invés da programação explícita. Ela permite que seja usada uma variedade de algoritmos para aprender iterativamente com os dados e melhorá-los, descrevê-los ou predizer resultados. Um modelo de *machine learning* é o nome dado ao *output* (em tradução livre: saída, resultado) obtido pelo treino de um algoritmo de *machine learning* com dados. Após o treinamento, podem ser fornecidos novos *inputs* (entradas) ao modelo, para que este calcule os respectivos *outputs*. Dentro da categoria de *machine learning*, existe, ainda, o *deep learning*, que nada mais é do que um método específico que incorpora redes neurais artificiais em camadas sucessivas para aprender iterativamente com os dados – o que é especialmente útil para que sejam encontrados padrões a partir de dados não estruturados.

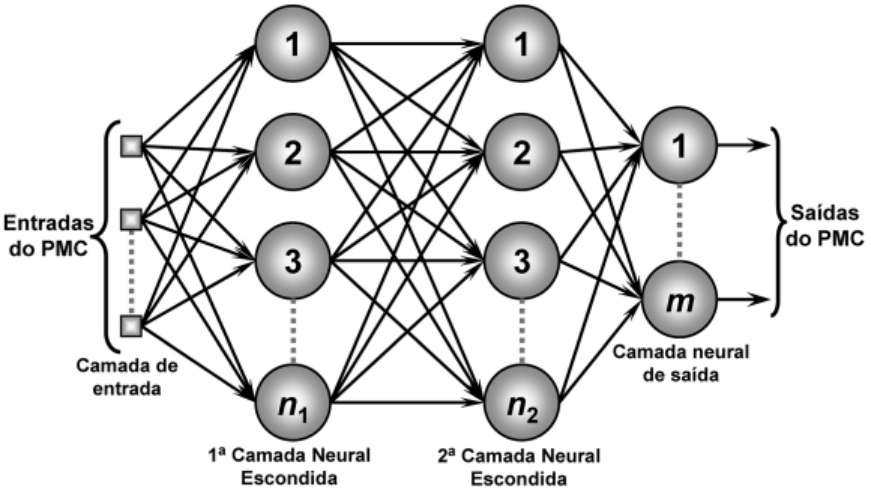


Figura 15 – Exemplo de Rede Neural: *Multi-Layer Perceptron*(CAMPBELL; NORMAN, 2012).

Ainda segundo Hurwitz e Kirsch (2018), redes neurais artificiais são tentativas de simular o modo como o cérebro humano interpreta problemas. Estas consistem em três ou mais camadas: uma camada de *input*, uma camada (ou mais) escondida e uma de *output*. Os dados são inseridos pela camada de *input* e então modificados na escondida e na de *output* baseado nos pesos e vieses aplicados a cada nó.

Existem infinitas arquiteturas possíveis para redes neurais, mas certas características são preferíveis para certos tipos de problemas. O fator que tem o seu impacto mais facilmente percebido é o número de camadas adicionadas, que conforme aumenta, causa também o aumento do custo computacional de treinar uma rede. Mas, como trata-se de um campo em desenvolvimento, ainda há vasta experimentação e frequentemente são sugeridas novas arquiteturas para problemas.

Uma rede neural pode ser explicada como uma grande função que funciona da seguinte maneira: com exceção da camada de *inputs*, o valor de cada neurônio é igual a uma soma ponderada de cada um dos pesos e valores dos neurônios da camada anterior somados a um viés. Isso pode ser matematicamente descrito por um produto matricial representado por $a^1 = \sigma(Wa^{(0)} + b)$ (e ilustrado na Fig. 16). Desse modo, os pesos e os valores armazenados por cada um dos neurônios são calculados, inclusive os da camada de *output*, que fornecem o resultado a ser interpretado com base em seus valores.

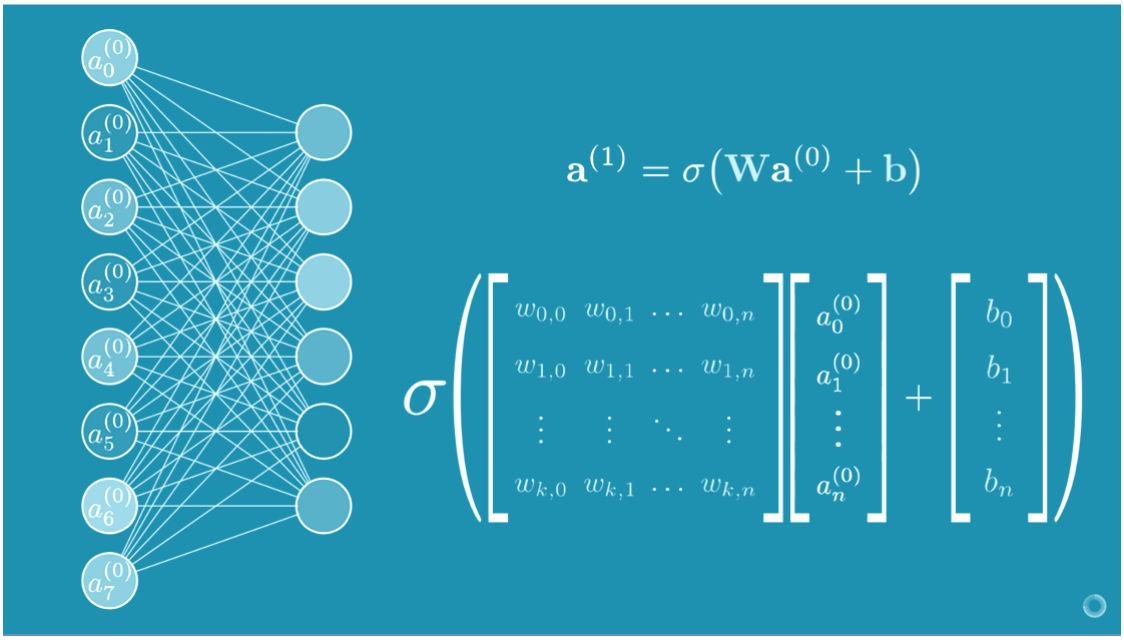


Figura 16 – Representação do cálculo do valor dos neurônios de uma rede neural (3BLUE1BROWN, 2017).

Para que esse processo ocorra, é necessário que seja escolhida uma função de ativação, que faz com que os valores de *inputs* possam ser trabalhados pela rede neural. Uma das funções mais comuns é a ReLU, que, em termos simplificados, "informa" se um determinado neurônio vai ser ativado ou não. Isto é feito considerando que se o valor retornado for maior que zero, seu valor será igual à sua função identidade, enquanto se for menor ou igual a zero, será considerado zero. Atualmente, esta função é mais comumente usada em redes neurais convencionais em comparação a outras mais tradicionais como a sigmoide, por ser mais eficiente em termos de custo computacional e convergir mais rapidamente. A Fig. 17 mostra alguns tipos de funções de ativações comuns hoje.

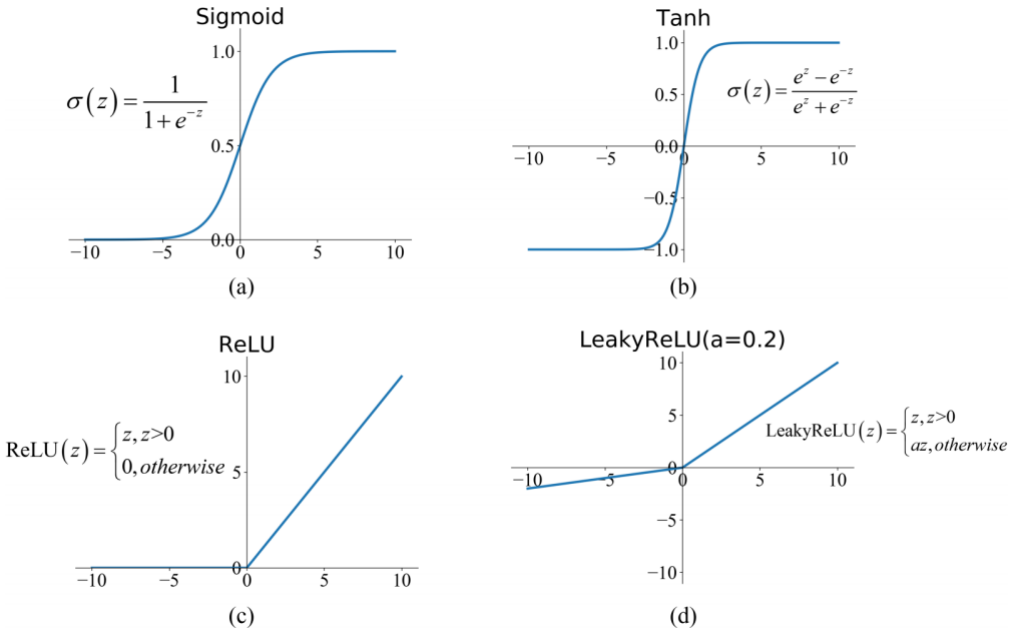


Figura 17 – Funções de ativação (FENG et al., 2019).

O que permite o "aprendizado" de uma rede neural é o seu treinamento, que pode ser de forma supervisionada ou não supervisionada. Para que isso aconteça, primeiro são gerados valores aleatórios que inicializam o processo, gerando um resultado de baixa qualidade. Em um treinamento supervisionado, esse *output* é então comparado com o que era esperado, e então é calculado um erro a partir de uma função de custo, também chamada de função de perda, ou *Loss Function*.

Para o cálculo da perda existem também diversas funções – algumas mais consolidadas, outras mais experimentais. Uma das funções clássicas é a do Erro Quadrático Médio, ou *Mean Squared Error* (MSE), definida como a soma dos quadrados das diferenças entre cada um dos *outputs* reais e dos esperados dividida pelo número de termos – conforme Eq. 2.31. Desse modo, quanto menor a perda final, mais próximo o resultado obtido é do esperado.

$$MSE = \frac{1}{n}\sum (y - \hat{y})^2 \tag{2.31}$$

Um método importante para a otimização das perdas no treino de um modelo é o *Stochastic Gradient Descent* (SGD), ou Descida de Gradiente Estocástica, método no qual cada iteração é feita com o intuito de encontrar o mínimo de uma função que mapeia os *inputs* e *outputs*, computando o gradiente e dando passos em direção ao seu negativo, conforme a Eq. 2.32. Nela, vale notar que o termo $\lambda \nabla_W R(W)$ refere-se à regularização, um conceito que busca tornar o modelo tão simples quanto possível, para que funcione melhor em dados de teste e não ocorra *overfitting*, ou seja, para que o modelo não aprenda a identificar somente os dados que foram usados para o treino ao invés das regras gerais.

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W) \tag{2.32}$$

Além disso, esse método é um dos que normalmente não calculam o gradiente de todos os dados de treino, mas apenas de amostras de um conjunto, chamadas de "*minibatch*" – que geralmente é estabelecido em uma potência de 2 (como 32, 64, 128), por convenção. Desse modo, é calculada uma estimativa representativa do gradiente completo.

Por fim, as iterações do treino são feitas por meio de um algoritmo chamado de *backpropagation*, ou "retropropagação", que ajusta os pesos das camadas anteriores aplicando a regra da cadeia para o cálculo do gradiente da função de custo com respeito a cada variável da rede.

Esse tipo de procedimento permite que aconteça a generalização dos "conceitos aprendidos", ou seja, a capacidade de reconhecer casos não vistos anteriormente e calculá-los com facilidade, e por isso é interessante o seu uso no problema das incrustações. As redes neurais de *feed forward*, ou seja, o tipo na qual as conexões não formam um ciclo, podem ser ilustradas de maneira simplificada como mostrado na Fig. 18.

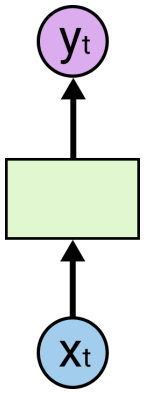


Figura 18 – Representação simplificada de uma rede neural *feed forward*.

Mas o problema analisado neste trabalho impõe algumas limitações que inviabilizam o uso de uma rede neural de *feedforward*, por se tratar de um processamento sequencial de dados, uma vez que esta categoria de arquitetura não é apta para lidar com informações temporais, além de todos os *inputs* serem processados de maneira independente.

Uma primeira ideia de solução seria realizar vários processamentos individuais com a mesma arquitetura, um para cada instante, de modo a obter uma série de dados. Entretanto, ainda assim os *outputs* seriam independentes dos resultados gerados pelos *inputs* anteriores.

Para solucionar este problema, existem as redes neurais recorrentes, chamadas RNNs. Trata-se de uma classe de redes na qual as conexões entre os nós formam um grafo direto ao longo de uma sequência temporal, permitindo um comportamento dinâmico.

Nela é possível processar dados sequenciais e de tamanhos variáveis com o uso de um estado interno dos nós da camada escondida, chamado de memória. Esse estado é usado no processamento dos *inputs*, como mostrado na Fig. 19, e é chamado de h_t. Para que isso aconteça, ele deve ser atualizado a cada instante em que a sequência é processada. Em outras palavras, esta relação de recorrência tem a função de incorporar ao modelo uma certa noção dos estados anteriores da sequência e a relação deles com os estados futuros.

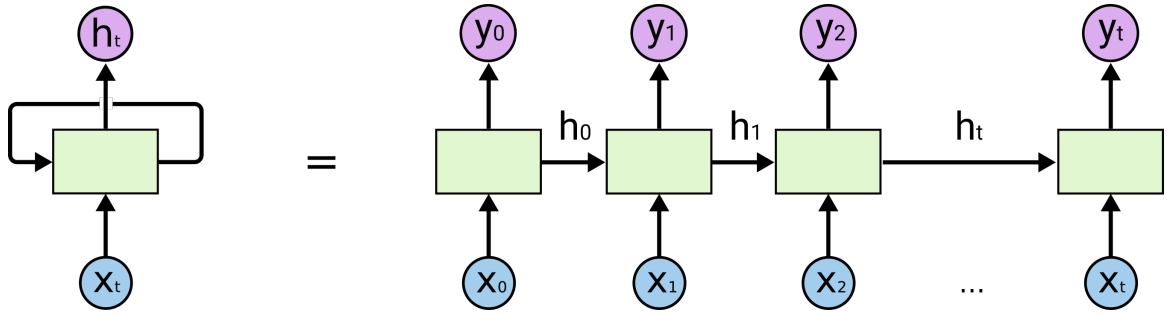
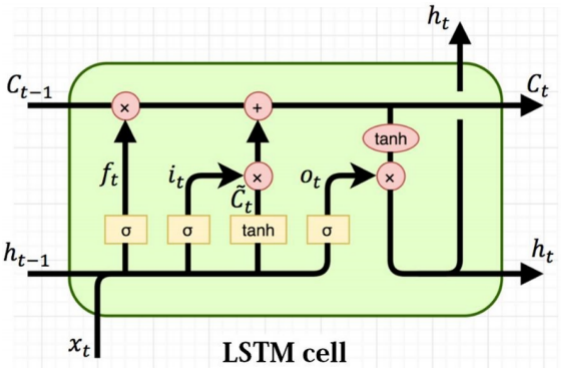


Figura 19 – Representação simplificada de uma rede neural recorrente.

O algoritmo de treino das RNNs é chamado *back propagation through time* e funciona como uma variação do algoritmo de *backpropagation* citado anteriormente. Nele, existe um problema que restringe a aplicação das redes neurais recorrentes convencionais: como as suas perdas são calculadas através de cada instante de tempo por meio de uma série de multiplicações de matrizes, existe o risco tanto da dissipação quanto da explosão do gradiente. Isto ocorre caso sejam multiplicados muitos valores menores que 1 ou muitos maiores que 1.

A arquitetura do tipo LSTM, desenvolvida por Hochreiter e Schmidhuber (1997), não possui esse problema, uma vez que o gradiente é calculado de maneira contínua. Nela, os estados internos são calculados conforme a Fig. 20. Por este motivo, essa foi a arquitetura escolhida.



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Figura 20 – Célula da LSTM (VARSAMOPOULOS et al., 2018).

3 Metodologia

3.1 Abordagem

A questão central para o desenvolvimento de um modelo preditivo que seja útil e prático em campo é a decisão de quais variáveis serão calculadas pela rede neural, tendo em mente o fim de se realizar futuramente uma análise de custo-benefício da limpeza dos equipamentos.

O melhor cenário seria um no qual a rede neural fosse treinada com dados de uma instalação real. Entretanto, segundo Bott (1995), como o desenvolvimento dos depósitos é lento, um único teste poderia durar semanas. Por exemplo, se fosse considerado um período otimista de 4 semanas para cada teste, em um ano seria possível coletar somente um conjunto de dados de 13 pontos, tornando inviável o processo para um estudo acadêmico.

Foram, portanto, gerados dados sintéticos por meio de um modelo teórico programado no EES que leva em conta as variações de temperatura externa de bulbo úmido e os parâmetros internos do *chiller*. Esse programa foi escolhido pela facilidade de obtenção das propriedades e de realização dos cálculos termodinâmicos. Essas simulações foram feitas de foma semelhante às feitas no trabalho de Meloni e Sousa (2019), mas aqui foram realizados aprimoramentos importantes.

Os dados gerados foram utilizados para o treino de redes neurais artificiais com diferentes parâmetros internos, com o objetivo de se obter um modelo capaz de reconhecer os padrões de incrustação quando confrontado com dados nunca vistos anteriormente. Aqui, foram separados os dados para treino e para teste, que permitem analisar a qualidade do modelo final. O modelo final permite sua utilização em aplicações reais e abre a possibilidade para que sejam criados novos modelos utilizando-se da estratégia aqui desenvolvida. A Fig. 21 resume em um fluxograma a metodologia utilizada.

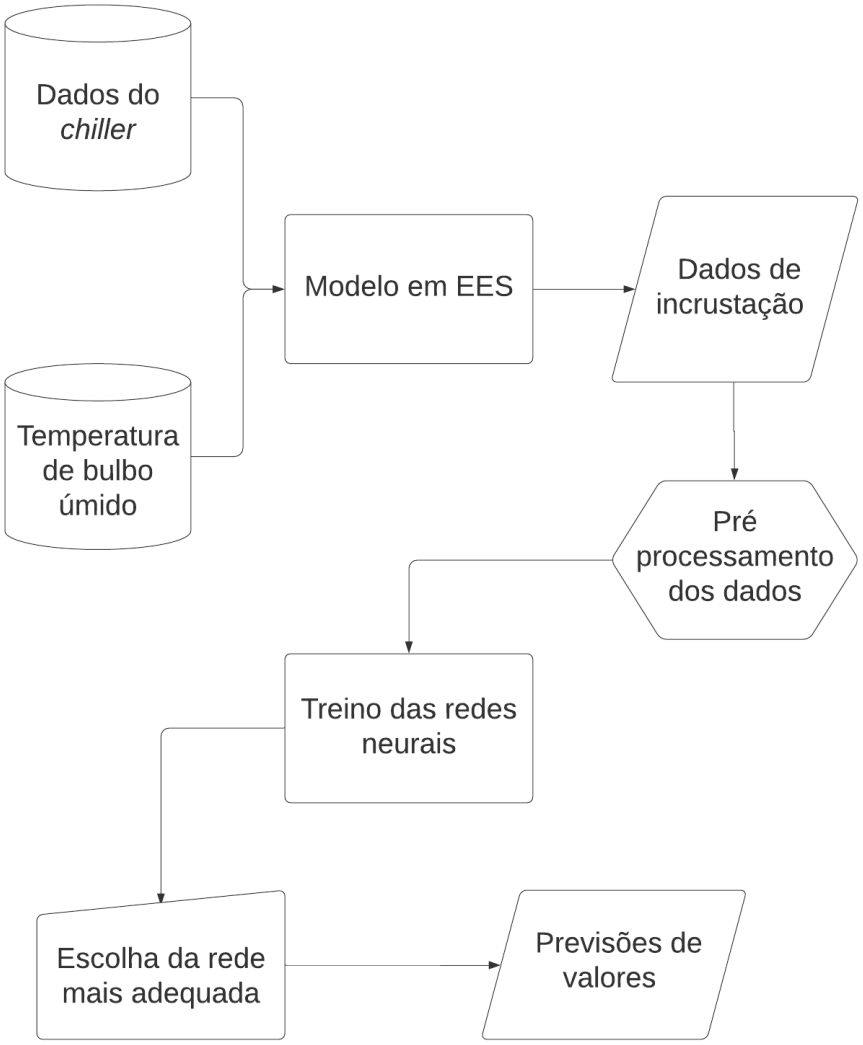


Figura 21 – Fluxograma da metodologia aplicada.

A linguagem utilizada para a programação da rede neural deste trabalho foi Python, com o auxílio das bibliotecas TensorFlow, Keras, NumPy, Pandas e scikit learn, que possuem ferramentas que auxiliam em grande medida o desenvolvimento da solução proposta. Além disso, o código foi executado no *Google Colab Notebook*, um ambiente interativo que permite sua execução sem que seja necessária nenhuma instalação ou configurações prévias no computador. Outra vantagem do *Colab* é a possibilidade do uso gratuito de GPUs (Unidades de Processamento Gráfico), executadas em servidores em nuvem da Google, o que permite um aumento considerável da capacidade de processamento.

3.2 Desenvolvimento do modelo matemático

O modelo matemático de um *chiller* virtual foi construído de forma que fosse possível resolvê-lo numericamente com o auxílio do *software* EES. A primeira condição de

contorno utilizada foi a temperatura de bulbo úmido de Brasília a cada hora, obtida por meio do *software Climate Consultant* 6 para um ano característico, calculado a partir das informações disponibilizadas pela estação meteorológica. A Fig. 22 mostra o perfil dessas temperaturas em função do tempo.

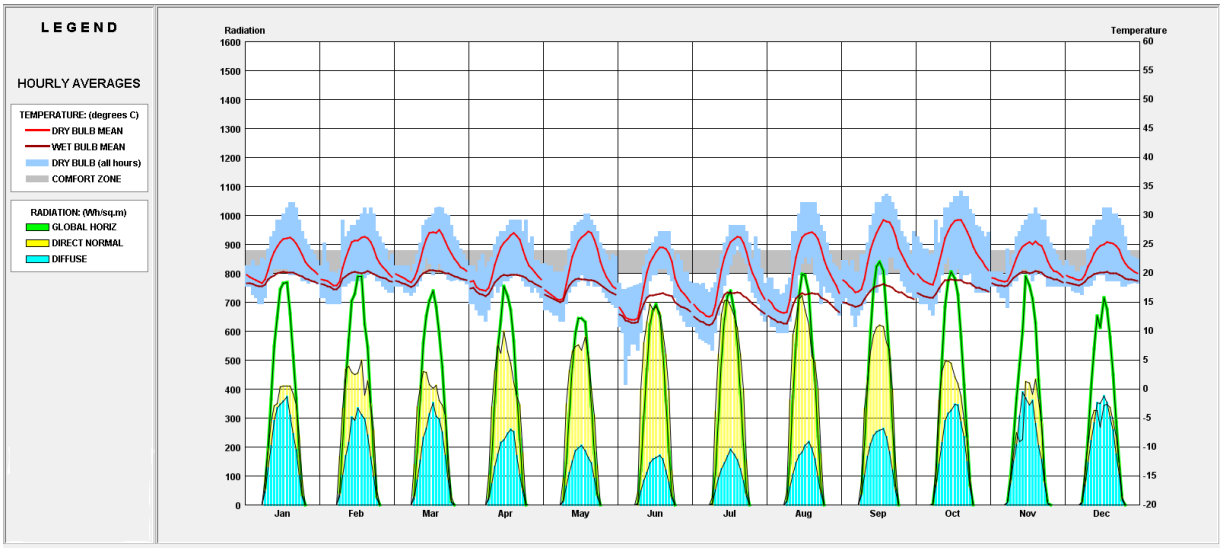


Figura 22 – Perfil de temperaturas de bulbo úmido, bulbo seco e radiação.

Além disso, foi considerado um *chiller* com dimensões e parâmetros de operação conforme as Fig. 23 e Fig. 24.

Descrição	Parâmetro	Valor
Número de tubos	N_t	500
Diâmetro interno	D_i	0,015 [m]
Diâmetro externo	D_o	0,017 [m]
Comprimento dos tubos	L	5 [m]
Condutividade térmica do cobre	k_cu	0,401 [kW/m*K]
Rugosidade relativa	RelRough	0,005

Figura 23 – Dados físicos do *chiller*.

Descrição	Parâmetro	Valor
Vazão de água no evaporador	Q_ev	0,0378 [m^3/s]
Vazão de água no condensador	Q_cd	0,031 [m^3/s]
Pressão da bomba	P_bomba	300 [kPa]
Temperatura na saída do condensador	T_o_cd	29,5 [C]
Temperatura de saída da água no evaporador	T_o_ev	7 [C]
Eficiência do compressor	eta_c	0,65
Vazão mássica do refrigerante	m_dot_ev	1,5 [kg/s]
Coeficiente global de transferência de calor * A	UA_ev	400 [W/K]
Capacidade de refrigeração	Q_dot_ev	450 [kW]

Figura 24 – Dados de operação do *chiller*.

Para a execução dos cálculos desejados, foram primeiramente declarados os valores dos *inputs* baseados nos dados citados previamente. Então, com base nos cálculos de troca de calor no condensador e no evaporador mostrados a seguir, foi feita a análise dos estados termodinâmicos em cada ponto do sistema, para que fossem calculadas as grandezas de temperatura T, pressão P, volume v, entropia s e entalpia h respectivamente associadas.

$$\dot{m}_{a,cd} = \rho_a Q_{cd} \tag{3.1}$$

$$\dot{m}_{a,ev} = \rho_a Q_{ev} \tag{3.2}$$

$$\dot{Q}_{cd} = \varepsilon_{cd} \dot{m}_{a,cd} c_a (T_3 - TBU) \tag{3.3}$$

$$\dot{Q}_{cd} = \dot{m}_{a,cd} c_a (T_{s,cd} - TBU) \tag{3.4}$$

$$\varepsilon_{cd} = 1 - \exp\left(\frac{-UA_{cd}}{\dot{m}_{a,cd} c_a}\right) \tag{3.5}$$

$$\dot{Q}_{ev} = \varepsilon_{ev} \dot{m}_{a,ev} c_a (T_{e,ev} - T_4) \tag{3.6}$$

$$\dot{Q}_{ev} = \dot{m}_{a,ev} c_a (T_{e,ev} - T_{s,ev}) \tag{3.7}$$

$$\varepsilon_{ev} = 1 - \exp\left(\frac{-UA_{ev}}{\dot{m}_{a,ev} c_a}\right) \tag{3.8}$$

Deste modo, é possível calcular também a potência do compressor, a capacidade de refrigeração do *chiller* e o COP.

$$\dot{Q}_{cd} = \dot{Q}_{ev} + \dot{W} \tag{3.9}$$

$$COP = \frac{\dot{Q}_{ev}}{\dot{W}} \tag{3.10}$$

Para o cálculo dos coeficientes de transferência de calor, utilizados conforme a Eq. 2.13, foram usadas as *procedures* internas *PipeFlow* e *Cond_horizontal_N_Cylinders* do EES. A *procedure PipeFlow* também informa a perda de carga em função do tempo.

Esse modelo foi, então, integrado a um modelo de progressão de incrustações, feito a partir da implementação do modelo descrito por Cremaschi e Wu (2015). Para isto, foi utilizada uma tabela paramétrica, que permite o cálculo de todos dados ponto a

ponto. Nela, foram calculados todos os valores que dependem da progressão da incrustação nos tubos. Ademais, foi feita a integração da Eq. 2.24 com o auxílio da função interna *"Integral"*, que utiliza um algoritmo preditor-corretor de segunda ordem e que define o tamanho do passo automaticamente.

$$D_{i,t} = D_i - 2\delta_t \tag{3.11}$$

$$\delta_t = R_f k_f \tag{3.12}$$

$$A_{i,t} = L\pi D_{i,t} N_t \tag{3.13}$$

$$u_t = \left(\frac{Q_{cd}}{N_t}\right) . \left(\frac{\pi D_{i,t}^2}{4}\right)^{-1} \tag{3.14}$$

3.3 Programação da rede neural

Para a programação e o treino de uma rede neural, é importante observar que não existe uma regra geral para a seleção dos parâmetros, e por este motivo é importante que seja realizado um certo número de testes. Para auxiliar este processo, foi utilizada a API (*Application Programming Interface*, ou Interface de Programação de Aplicações) TensorBoard, por ser uma ferramenta que possibilita uma representação visual dos desempenhos de testes com diferentes métricas, como visualização dos histogramas de pesos.

Sabendo que o problema em questão trata-se de uma série temporal, foi adotada uma arquitetura do tipo LSTM Bidirecional – a adição da bidirecionalidade permite que o modelo interprete os *inputs* em duas direções, uma do início para o fim e uma do fim para o início, de forma a captar melhor o contexto dos dados. A função de ativação escolhida para esse tipo de arquitetura foi a "tanh" (tangente hiperbólica). Esta função não é explicitada no código em Python por ser a padrão na biblioteca Keras para redes LSTM.

A otimização foi feita pelo sofisticado algoritmo Adam (*Adaptive Moment Estimation*). Segundo Kingma e Ba (2014), este método é computacionalmente eficiente, possui pouco requerimento de memória e é adequado para problemas com muitos dados ou parâmetros. Seu funcionamento se dá pela computação de uma taxa de aprendizado adaptativa para cada parâmetro. Ele é comumente descrito como uma forma de junção de dois outros algoritmos: o *gradient descent with momentum* (o algoritmo de descida de gradiente com a adição de maior velocidade de otimização) e o RMSprop, que permite a adaptatividade da taxa de aprendizado. Esta taxa é normalmente chamada de *learning*

rate e define o tamanho de cada passo dado pelo algoritmo de treino. Ademais, a função de perda escolhida foi a do Erro Quadrático Médio, descrita no capítulo anterior.

A escolha do número de nós foi feita a partir do método de tentativa e erro, levando em conta que é necessário que o modelo possua um número de nós altos o suficiente para a modelagem correta, mas baixos o suficiente para que seja garantida generalização e não ocorra *overfitting*. Além disso, este parâmetro também impacta diretamente o custo computacional. Após a camada escondida, foi adicionada uma taxa de *dropout* de 20%, servindo a função de regularização da rede. Este método ignora aleatoriamente alguns *outputs* da camada, fazendo com que ela seja temporariamente tratada como se possuísse um número diferente de nós. Ao fim, foi inserida uma camada totalmente conectada com apenas um nó, servindo como *output*.

Dentre os vários *inputs* e *outputs* disponíveis no modelo matemático realizado anteriormente, foram escolhidos para o treino o R_f (como *output*), e o tempo, a temperatura de bulbo úmido, a temperatura de condensação, a condutância, a pressão da bomba e o comprimento dos tubos como *inputs*. A escolha foi feita desta maneira por se tratar de dados plausíveis de serem obtidos em uma instalação real, conferindo maior utilidade prática do modelo final.

O treino foi realizado com 10.000 pontos separados em dados de treino (90%) e de teste (10%), também chamados de validação. Em seguida foi feito o pré-processamento e o treino de fato. Ao final de todo o código, foi calculada a predição de R_f para cada instante de tempo.

4 RESULTADOS E DISCUSSÕES

4.1 Do modelo matemático

O modelo do *chiller* virtual foi executado conforme explicado no capítulo anterior, com os dados de temperatura de bulbo úmido a cada hora, computando iterativamente os parâmetros que são afetados pela incrustação nos tubos do condensador. Para fins de comparação, foi também executada uma rotina que considera o caso dos tubos limpos, de modo a possibilitar a visualização do efeito causado pela incrustação.

Os resultados podem ser visualizados nas figuras a seguir. A progressão da incrustação foi analisada a partir do parâmetro R_f, que apresenta um crescimento conforme o gráfico mostrado pela Fig. 25. As oscilações desta curva se devem às variações a cada instante da velociade da água, do diâmetro interno e do número de Reynolds, que afetam diretamente os demais parâmetros envolvidos no fenômeno da incrustação.

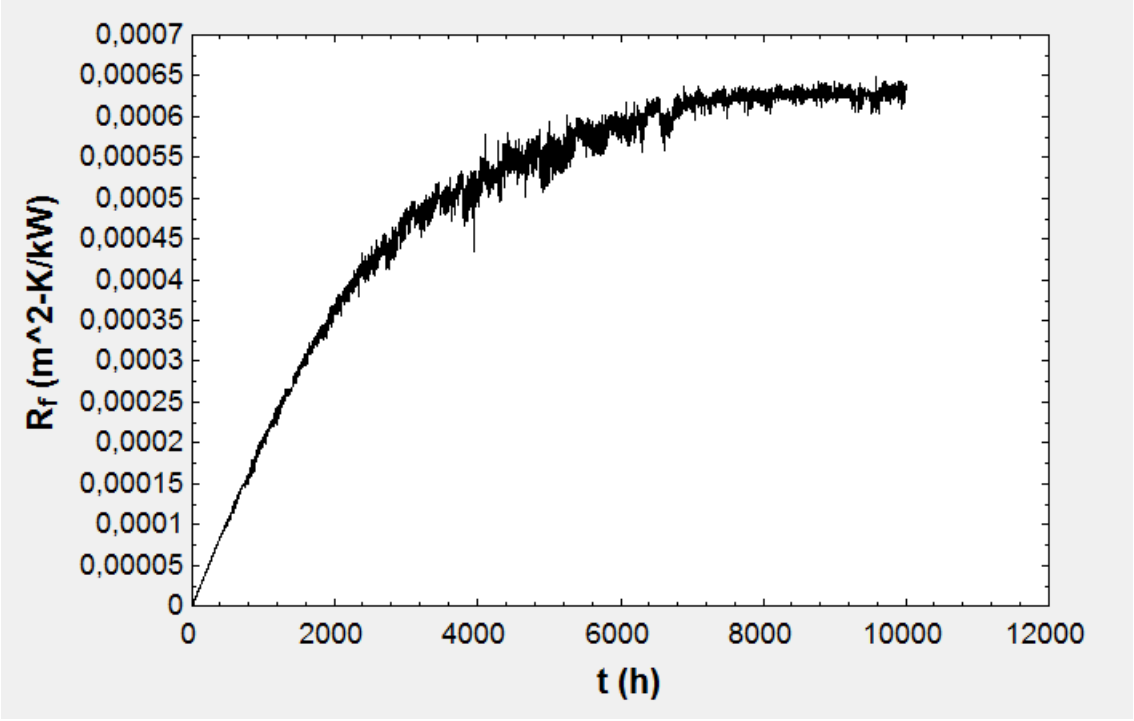


Figura 25 – Resistência térmica causada pela incrustação em função do tempo.

Como a espessura da camada depositada aumenta com o tempo, o diâmetro interno

e a área total interna também variam, decaindo conforme mostrado pelas Figs. 26 e 27. Como consequência, a velocidade no interior dos tubos aumenta, como na Fig. 28; estes efeitos acarretam o aumento da perda de carga, conforme Fig. 29.

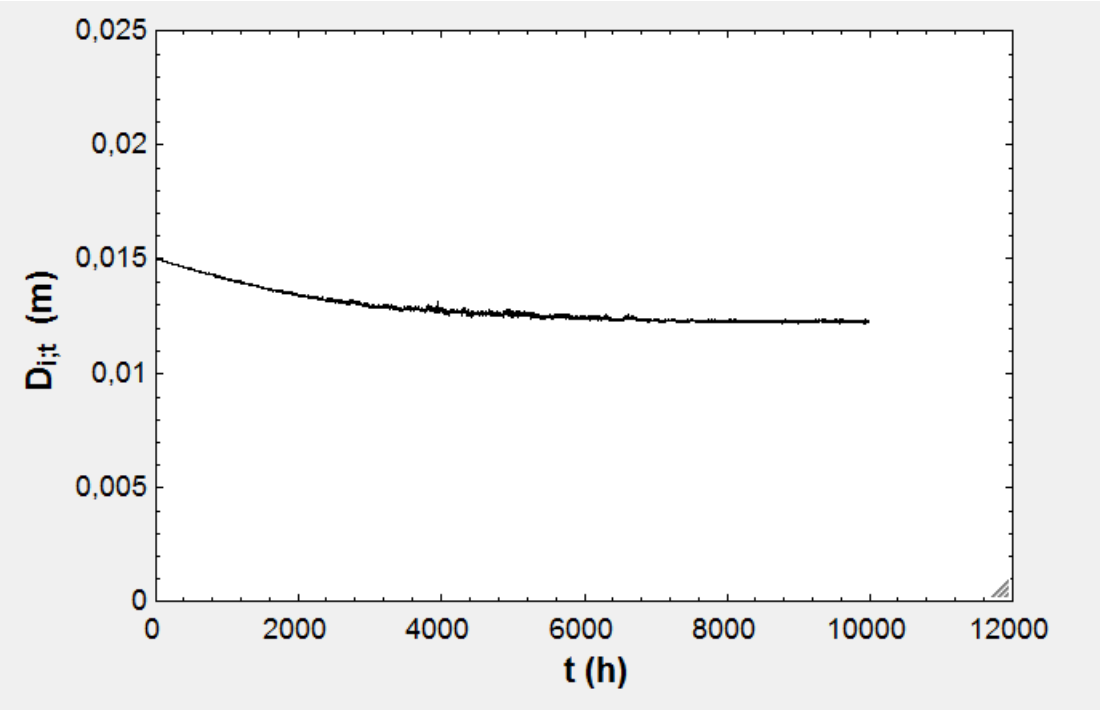


Figura 26 – Diâmetro interno em função do tempo.

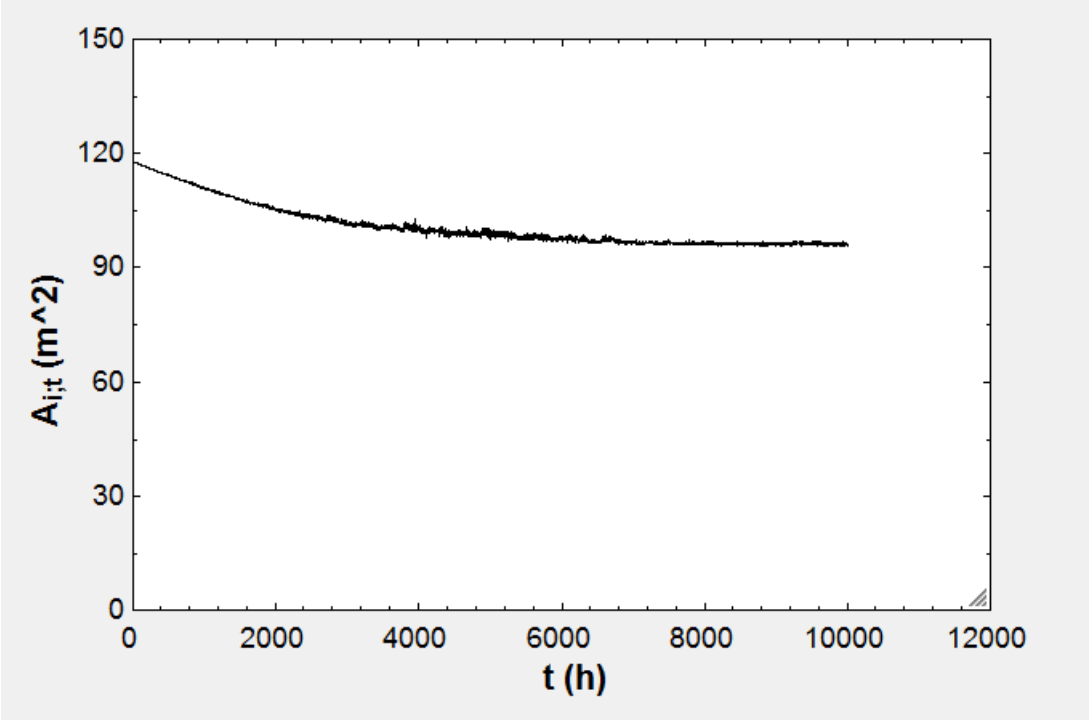


Figura 27 – Área interna dos tubos em função do tempo.

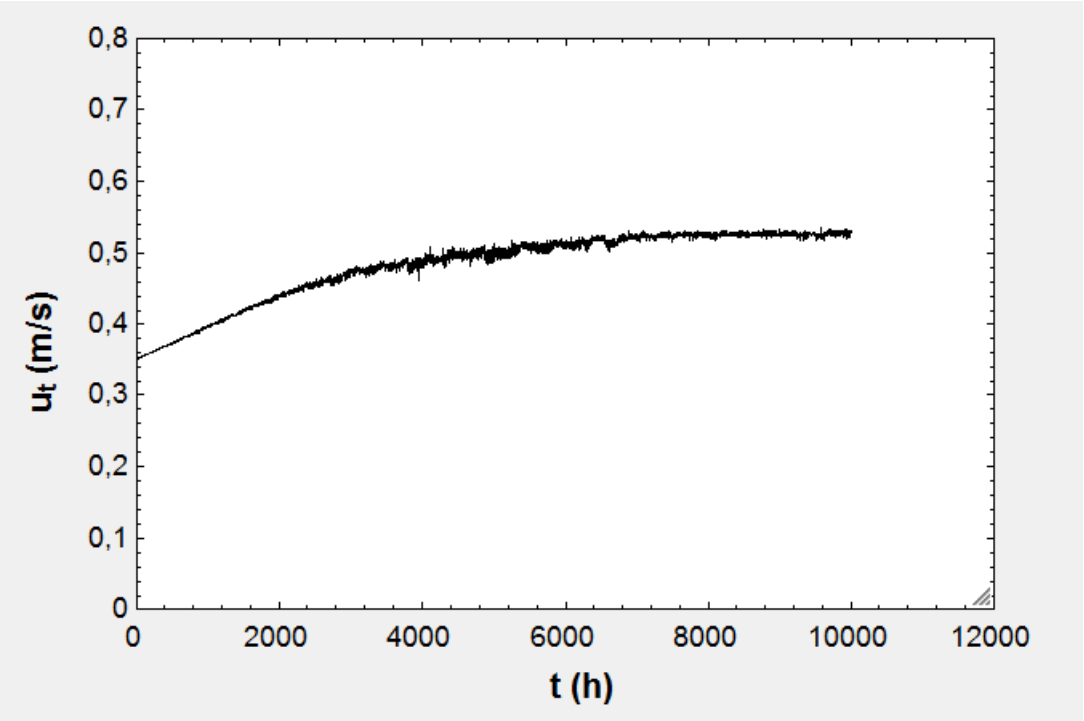


Figura 28 – Velocidade da água no condensador em função do tempo.

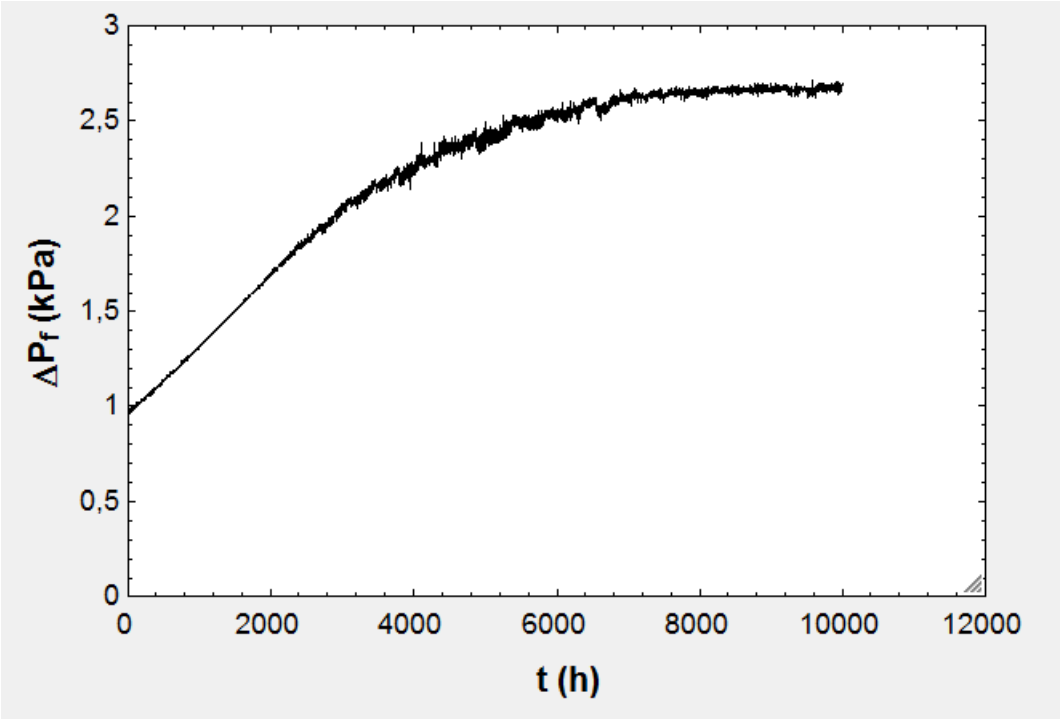


Figura 29 – Perda de carga em função do tempo.

Para melhor visualização dos efeitos causados pelas incrustações, os gráficos da temperatura de condensação e condutância foram traçados em sobreposição com os respectivos resultados em um caso sem incrustação. Além disso, foi feita a média móvel de 100 dias para que a curva fosse suavizada e, consequentemente, permitisse uma interpretação mais clara.

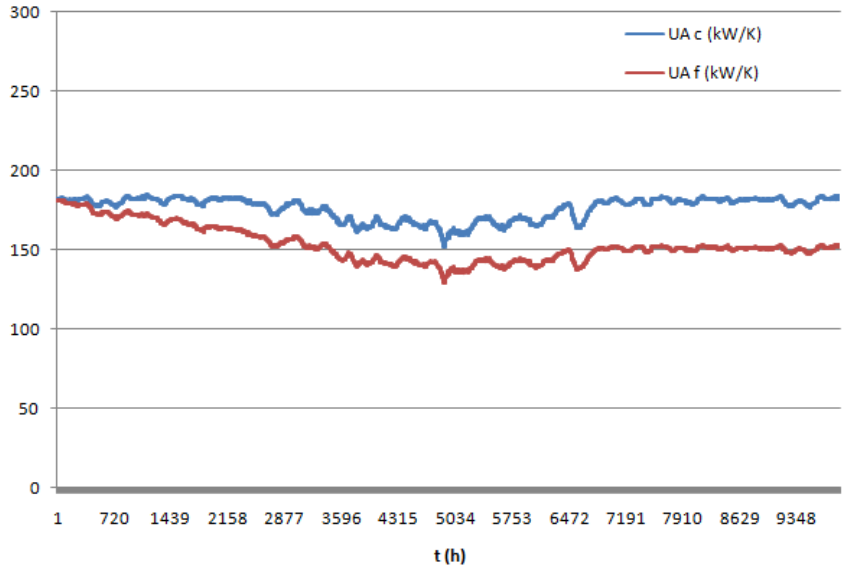


Figura 30 – UA do condensador em função do tempo para os dois casos.

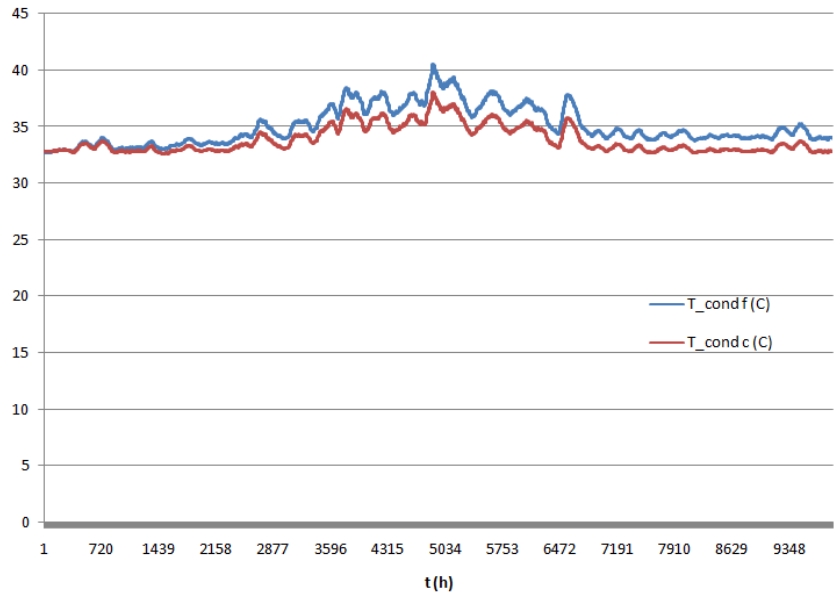


Figura 31 – Temperatura de condensação em função do tempo para os dois casos.

Como esperado, as incrustações causaram a redução no produto UA do condensador, o aumento da temperatura de condensação e o aumento da perda de carga nos tubos.

4.2 Das redes neurais

Para que os dados gerados pelo modelo matemático pudessem ser utilizados para o treino da rede neural, foi feito o código da rede LSTM com uma estrutura final conforme mostrado pela Fig. 32, esta feita com o auxílio do TensorBoard, e pelo Apêndice B.

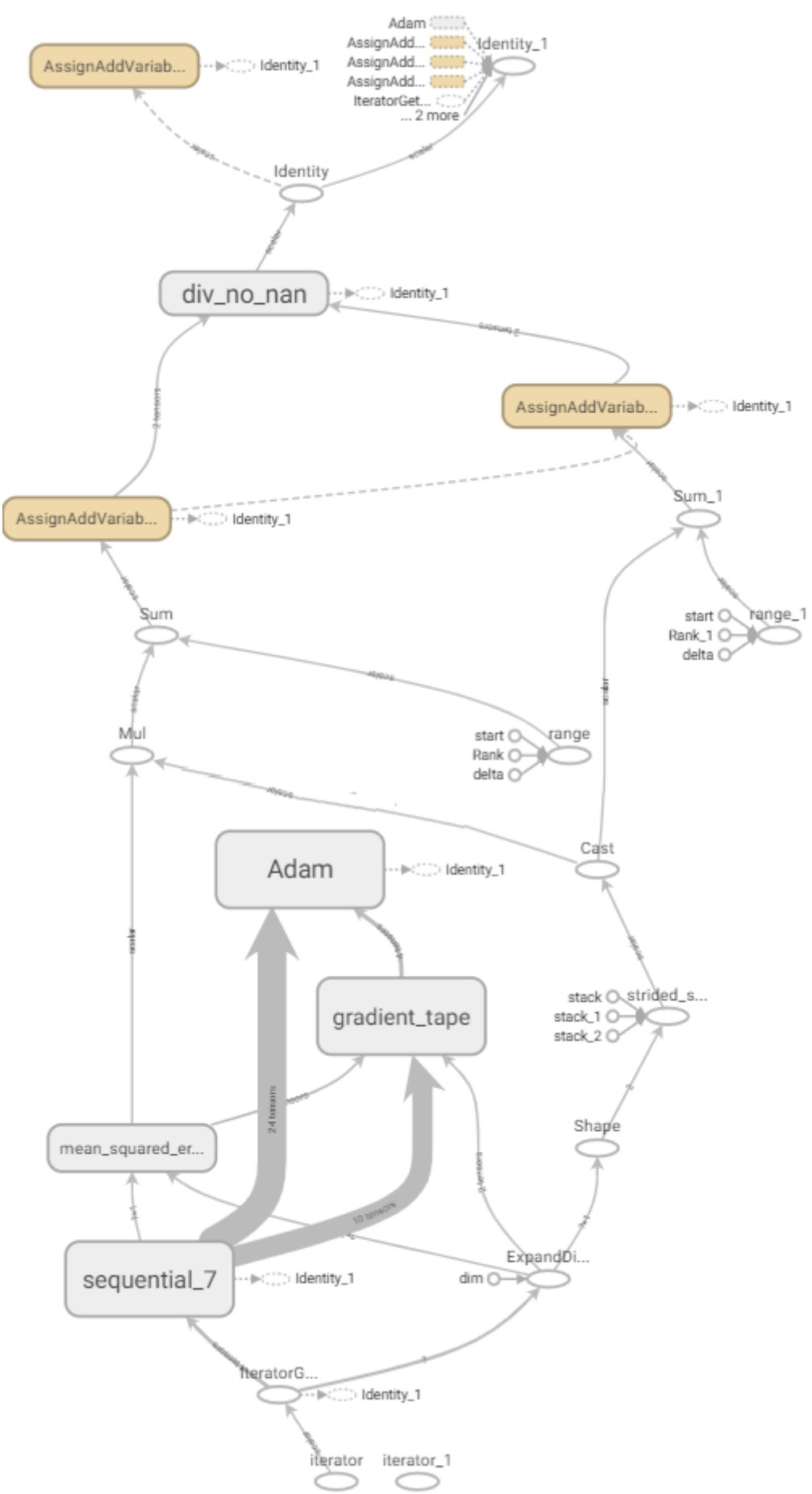


Figura 32 – Estrutura do código.

Em seguida, diversas simulações de treino foram feitas seguindo ajustes progressivos dos hiper-parâmetros. Para isso, foram primeiramente testadas variações bruscas (com o fim de obter uma direção adequada de otimização) e, em seguida, os ajustes foram realizados em escala menor. A principal ferramenta utilizada para a análise da qualidade dos parâmetros adotados foi a curva de perda, que traça a relação entre aprendizado (no eixo y) e a experiência (no eixo x). As variações adotadas para os ajustes finos foram a da taxa de aprendizado (0,00001, 0,0001 e 0,001) e a do número de nós (128, 256 e 512), todas realizadas para um número de 100 épocas. As Fig. 33 e Fig. 34 mostram as funções de perda resultantes de cada uma das iterações dos hiper-parâmetros (é importante notar que cada um dos números que iniciam os nomes dos testes indicam metade do número de nós da rede, por ela ser bidirecional).

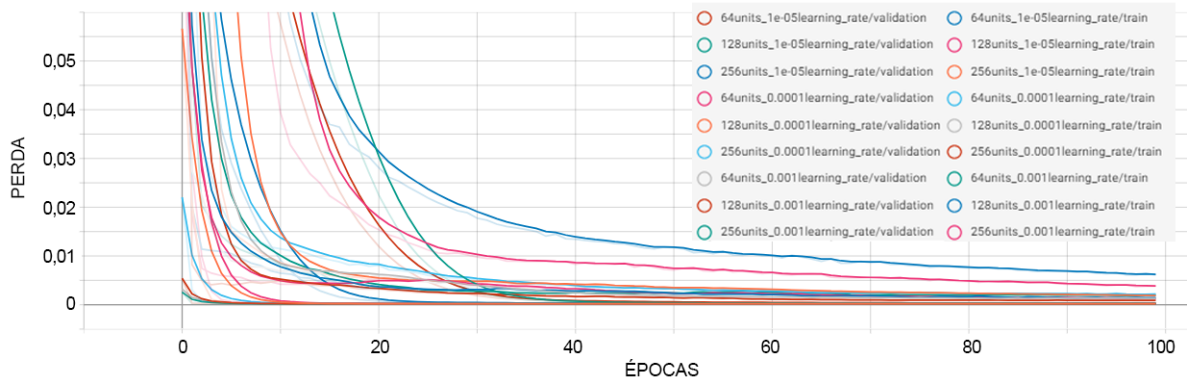


Figura 33 – Função de perda para os diferentes testes com suavização de 60%.

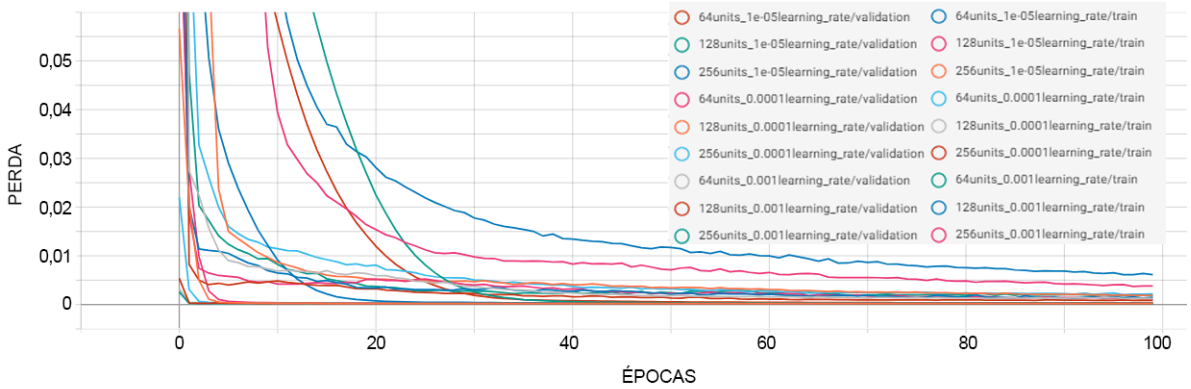


Figura 34 – Função de perda para os diferentes testes sem suavização.

Como os gráficos acima apresentam quantidades altas de informações e difícil leitura, é interessante que as curvas sejam separadas em duas categorias: as de treino e as de validação, como mostrado a seguir. A Fig. 35 mostra a curva de perda para os dados de treino; já o desempenho nos dados de validação é mostrado pela Fig. 36. Os eixos horizontais foram restritos a 60 épocas, também para melhor visualização.

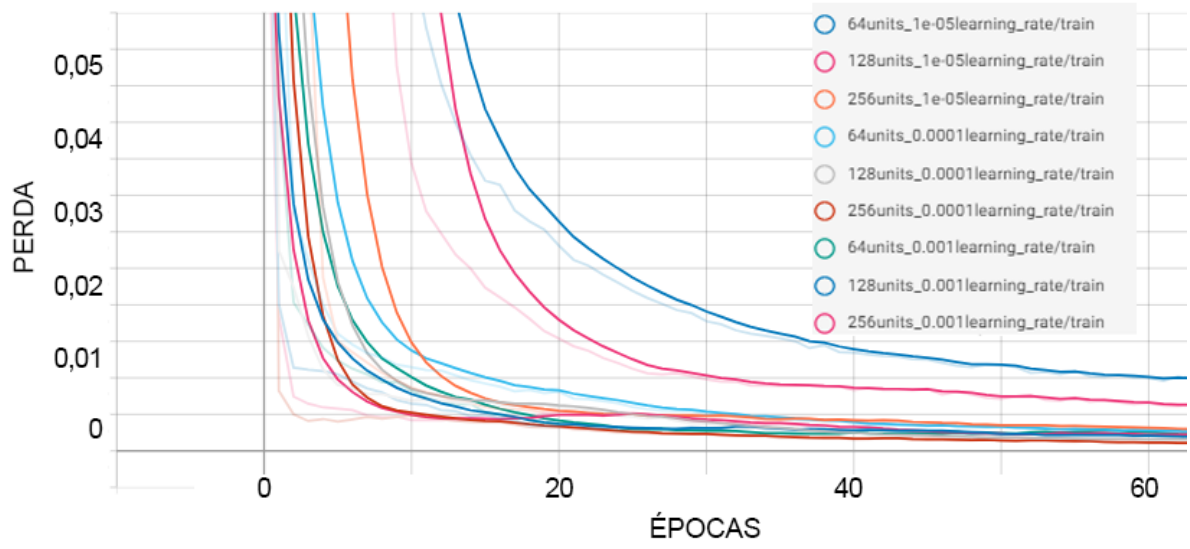


Figura 35 – Curva de perda em função das épocas para os dados de treino (amplificada).

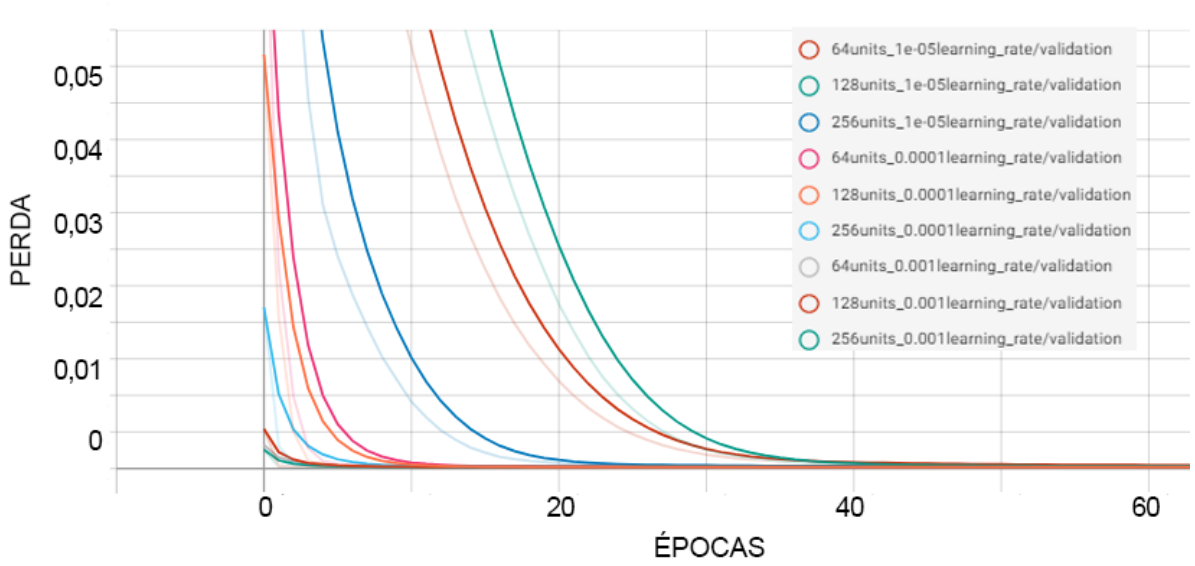


Figura 36 – Curva de perda em função das épocas para os dados de validação (amplificada).

Vale notar que os dados de validação são mais importantes para a boa compreensão do modelo do que os de treino, uma vez que aquele trata da diferença entre os valores calculados e os dados de validação, ou seja, que não foram usados para o treino. Além disso, as figuras anteriores ilustram que, como esperado, o aumento do número de nós tende a apresentar um resultado melhor, mas só até certo ponto, uma vez que aumenta a possibilidade de *overfitting*. Apesar da variação da taxa de aprendizado ter sido conservadora, é possível observar que, caso este parâmetro seja muito alto, os resultados oscilarão e talvez sequer convirjam – e se for muito baixo, o aprendizado pode não ser suficiente.

Além disso, é possível visualizar a distribuição dos pesos por meio de histogramas, que permitem uma visualização mais detalhada do que acontece no interior de cada uma das redes. Os gráficos a seguir mostram as distribuições dos vieses (nas imagens com as legendas "*bias*") e pesos (nas imagens com as legendas "*kernel*") da camada totalmente conectada para cada um dos casos calculados. Cada camada do gráfico mostra a distribuição de frequência dos valores em um único passo. O eixo x representa os valores e o eixo y representa a frequência. Para os histogramas dos pesos, é importante notar que, tanto uma distribuição que se assemelhe a uma inicialização aleatória do pesos, quanto valores muito próximos de zero costumam ser indícios de baixo aprendizado da rede.

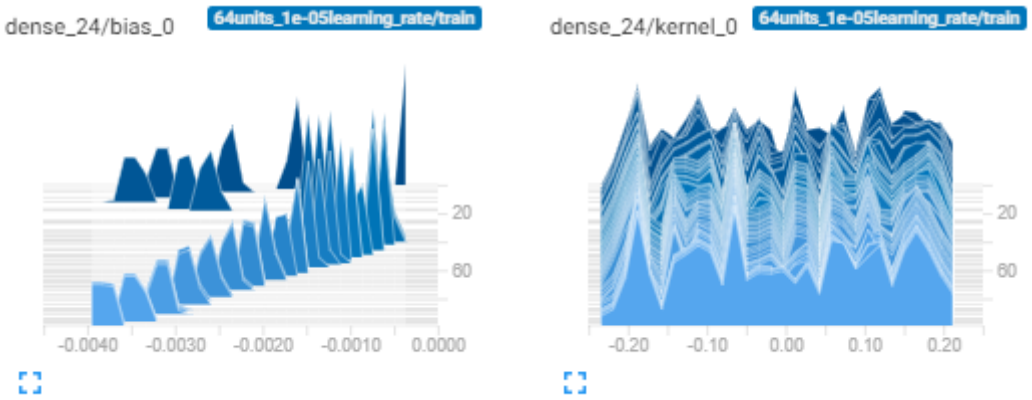


Figura 37 – Histograma para 128 nós e taxa de aprendizado de 0,00001.

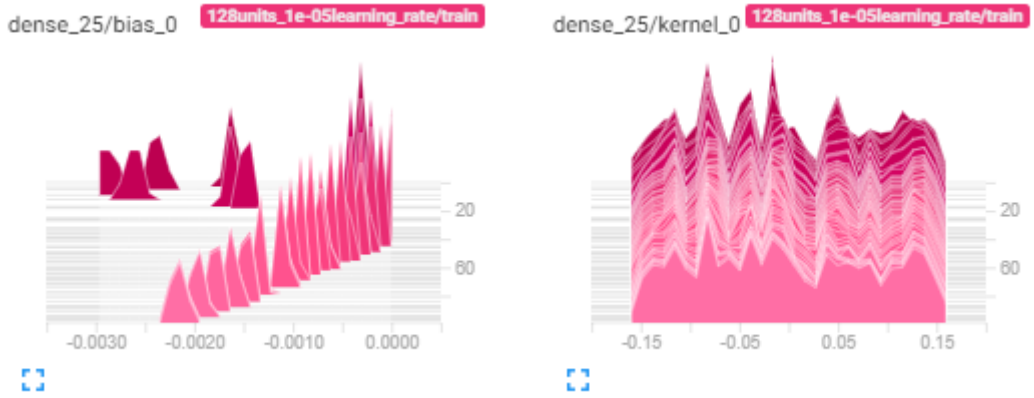


Figura 38 – Histograma para 256 nós e taxa de aprendizado de 0,00001.

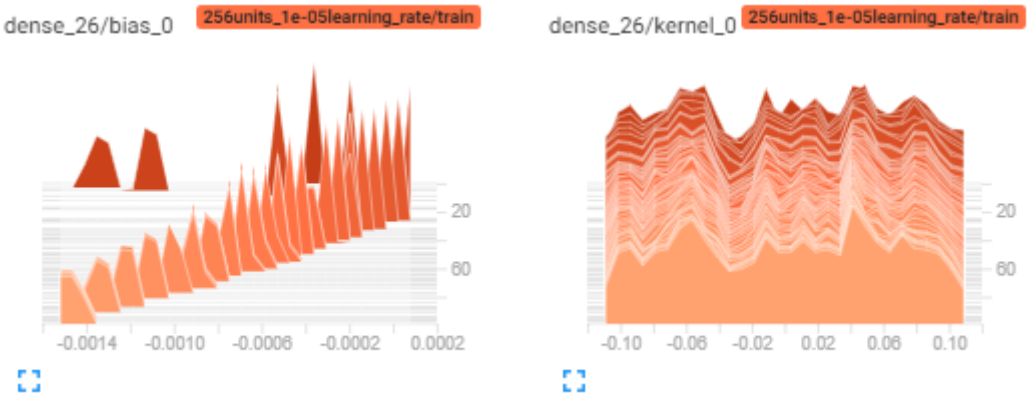


Figura 39 – Histograma para 512 nós e taxa de aprendizado de 0,00001.

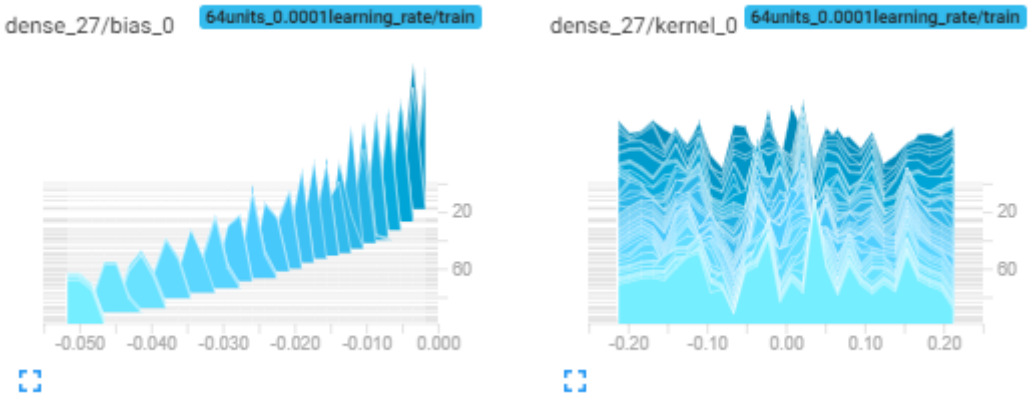


Figura 40 – Histograma para 128 nós e taxa de aprendizado de 0,0001.

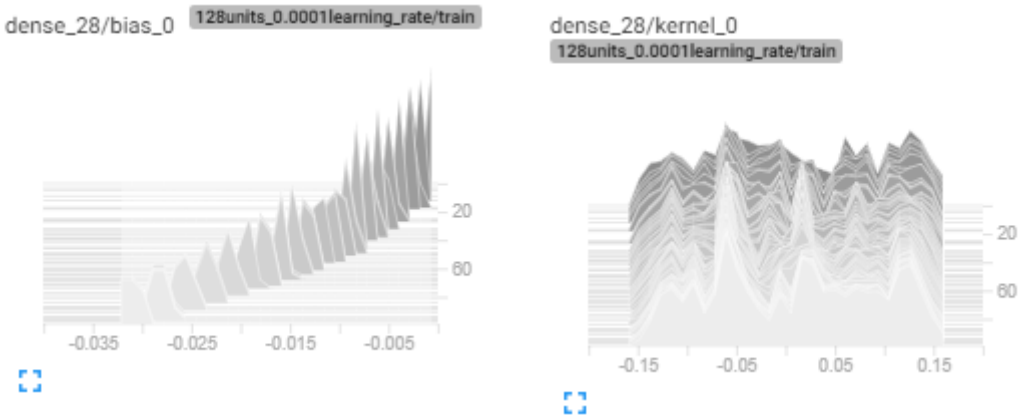


Figura 41 – Histograma para 256 nós e taxa de aprendizado de 0,0001.

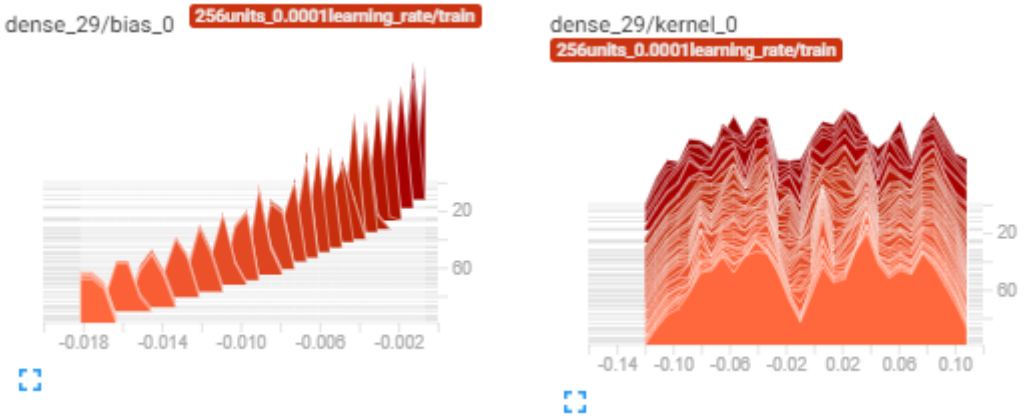


Figura 42 – Histograma para 512 nós e taxa de aprendizado de 0,0001.

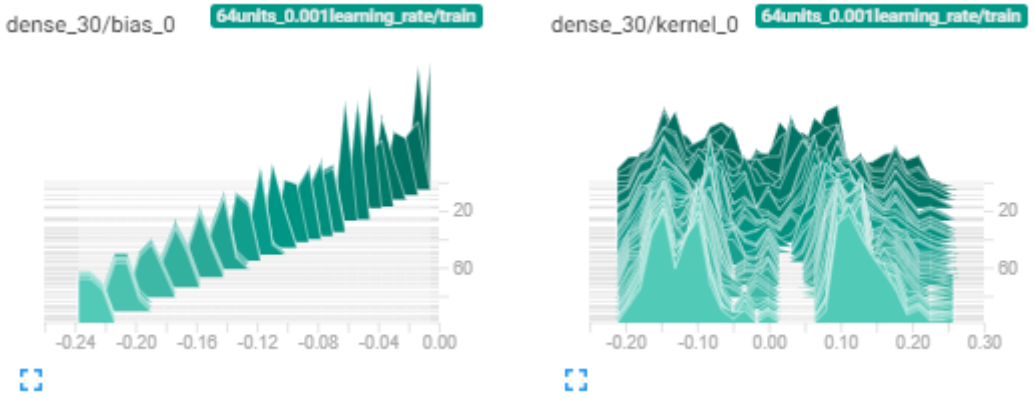


Figura 43 – Histograma para 128 nós e taxa de aprendizado de 0,001.

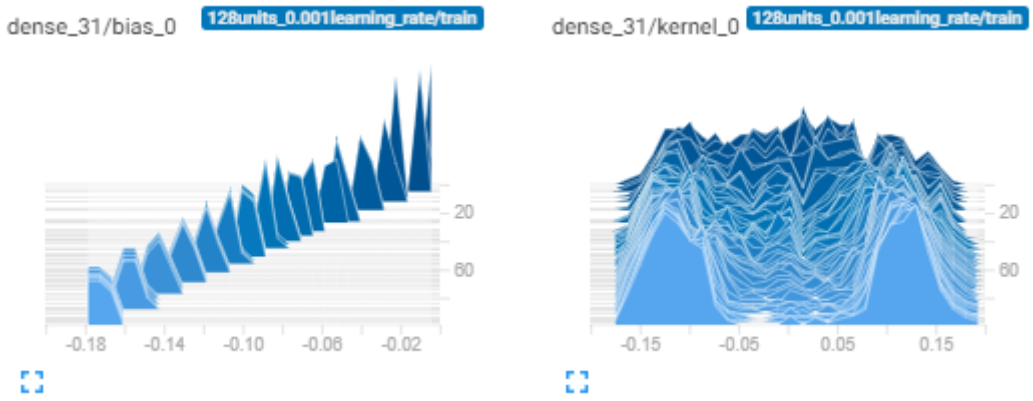


Figura 44 – Histograma para 256 nós e taxa de aprendizado de 0,001.

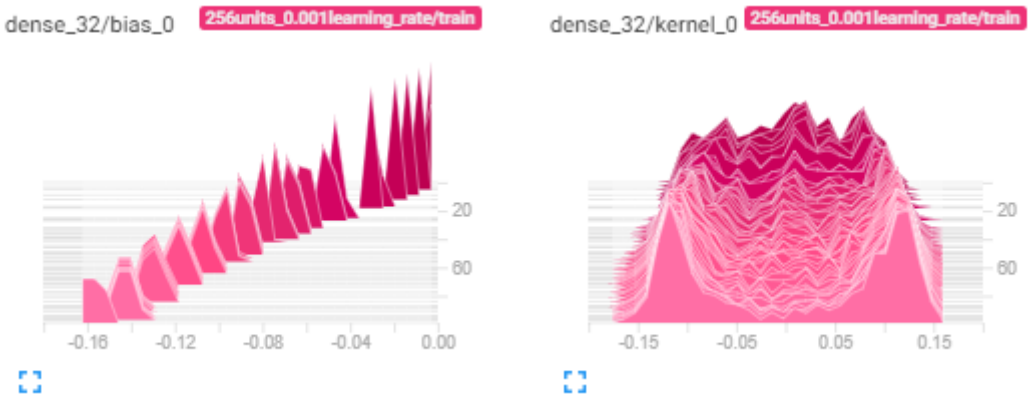


Figura 45 – Histograma para 512 nós e taxa de aprendizado de 0,001.

A partir das informações obtidas com os gráficos mostrados anteriormente, foi escolhida como final a arquitetura com 256 nós e taxa de aprendizado de 0,0001 treinada por 30 épocas, por fornecer um bom resultado e não apresentar grandes indícios de *overfitting*. Invertendo o processamento dos dados feitos ao início do código, é possível traçar os valores de *outputs* preditos por essa rede, como mostrado pelas Fig. 46 e Fig. 47.

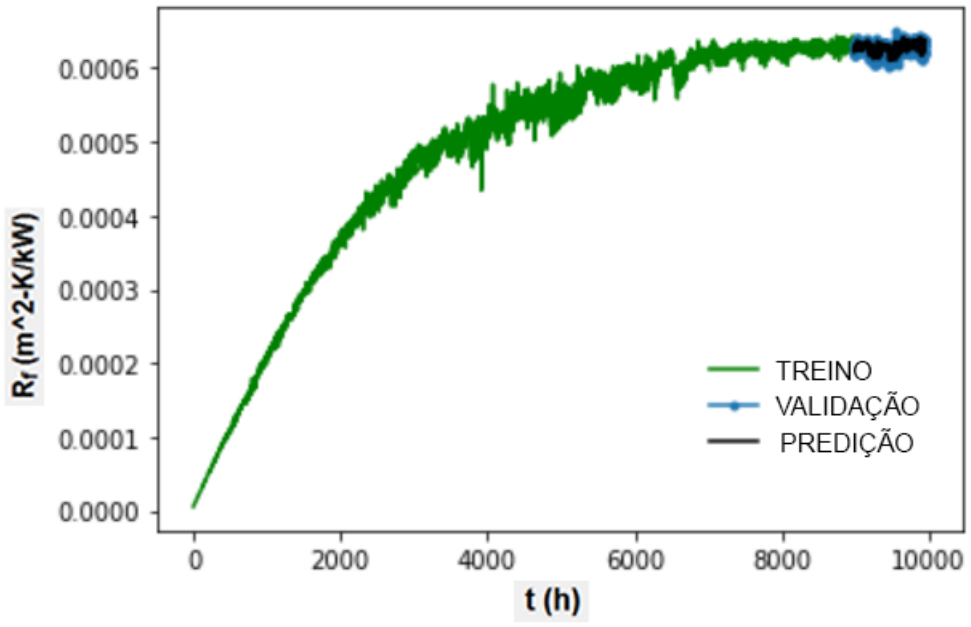


Figura 46 – Visualização sem *zoom*, incluindo dados de treino.

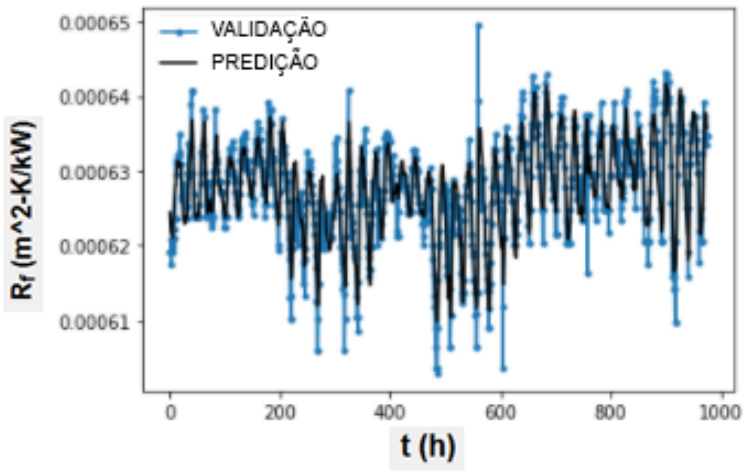


Figura 47 – Superposição com *zoom* dos valores preditos aos dados de teste.

A diferença entre os valores gerados pelo modelo matemático e os calculados pela rede neural pode ser visualizada na Fig. 48. Nela, a reta tracejada central é a reta $y = x$, traçada para referência.

Figura 48 – Dispersão dos valores preditos e de validação de Rf.

A Fig. 49 apresenta um resumo da relação entre o número de unidades da rede neural, a taxa de aprendizado, a perda nos dados de treino a cada época e a perda nos dados de validação a cada época, possibilitando uma visualização mais precisa do impacto de cada um dos parâmetros adotados. Sua legenda é apresentada na Fig. 50.

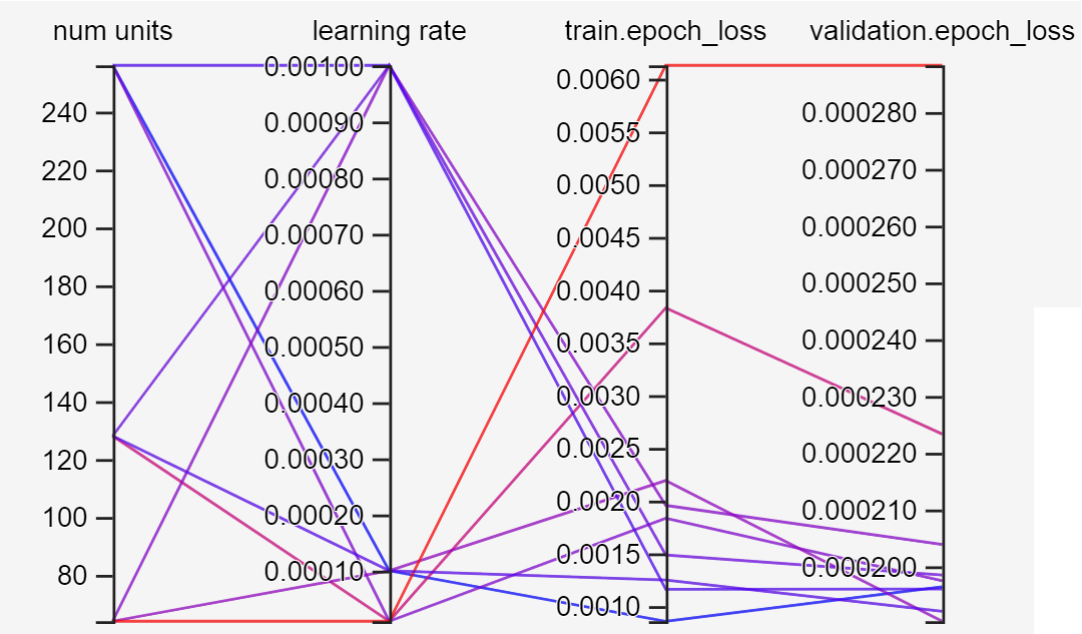


Figura 49 – Tabela de hiper-parâmetros.

nº unidades	taxa de aprendizado	perda / época - treino	perda / época - validação
256	0,0001	0,00085714	0,00019640
128	0,00001	0,0038270	0,00022330
64	0,001	0,0019551	0,00020384
256	0,00001	0,0018335	0,00019747
64	0,0001	0,0021903	0,00019034
128	0,001	0,0014844	0,00019846
64	0,00001	0,0061269	0,00028824
128	0,0001	0,0012482	0,00019209
256	0,001	0,0011597	0,00019602

Figura 50 – Legenda da tabela de hiper-parâmetros.

Para o uso desse modelo em aplicações diferentes da estabelecida neste trabalho, pode ser de grande utilidade o método de *transfer learning*. Segundo Goodfellow, Bengio e Courville (2016), a transferência de aprendizado e adaptação de domínio referem-se à situação na qual o que foi aprendido em um contexto é aproveitado para aumento da generalização em outro contexto. Dito de outro modo, com ele, é possível que se use uma rede neural previamente treinada em uma aplicação que ela poderia não estar preparada, realizando outro treino "por cima" do anterior. Desse modo, a rede neural poderia se adaptar melhor a casos mais variados de incrustação.

5 CONCLUSÃO

5.1 Contribuições do trabalho

Este trabalho baseou-se essencialmente na literatura já consolidada sobre trocadores de calor, no modelo de progressão de incrustação de Cremaschi e Wu (2015) e no seu desenvolvimento feito por Meloni e Sousa (2019), explorando um campo ainda pouco estudado, que é a aplicação do aprendizado de máquina para problemas de engenharia mecânica.

Os principais resultados obtidos foram o desenvolvimento de um modelo de *chiller* virtual, a avaliação dos impactos da incrustação no desempenho deste, a criação e treino da rede neural LSTM Bidirecional e a preparação de um caminho que possa ser usado como base por outros estudos e por aplicações em instalações reais. Os resultados foram condizentes com os esperados e mostram que este tipo de abordagem é viável.

5.2 Sugestões para trabalhos futuros

A primeira sugestão para trabalhos futuros é o teste em instalações reais da rede neural gerada. É possível que estes façam uso de um modelo *online*, que continuamente se adapte conforme novos dados forem coletados, como mostrado por Sahoo et al. (2017). Isto seria de grande utilidade, uma vez que o modelo seria beneficiado fortemente do treino contínuo pela obtenção de dados reais simultâneos.

Além disso, é possível que o modelo aqui desenvolvido seja adaptado a novos casos por meio do método da transferência de aprendizado. Desse modo, haveria um potencial ainda maior de predições acuradas.

Outra direção interessante seria o cruzamento de outros modelos apresentados na literatura com os resultados obtidos por este trabalho. É possível, ainda, que seja implementada aos cálculos a decisão de qual o momento ideal para que um equipamento seja parado e limpo. Ademais, cabe também o aprimoramento de algumas hipóteses simplificadoras no modelo do *chiller* virtual, como a adição do efeito do tratamento da água usada no condensador, o efeito causado pelos períodos sem funcionamento e a introdução de outros tipos de incrustação.

Pode também ser feito o estudo do impacto da consideração de outros parâmetros do sistema físico pela rede neural. Um deles, por exemplo, é a inserção de dados de categoria, como o material da tubulação ou o fabricante. Por fim, é interessante que sejam feitos novos testes que experimentem com diferentes tipos de arquiteturas.

Referências

3BLUE1BROWN. But what is a neural network? 2017. Disponível em: <https://www.youtube.com/watch?v=aircAruvnKkab_channel=3Blue1Brown>. Citado 2 vezes nas páginas iv e 23.

AGUIAR, R. A. Z.; LEAL, V. A. Modelagem de referência para ar condicionado de janela utilizando o software engineering equation solver. 2014. Citado na página 6.

AWAD, M. M. *Fouling of heat transfer surfaces*. [S.l.]: INTECH Open Access Publisher, 2011. Citado na página 17.

BEMROSE, C.; BOTT, T. Theory and practice in gas-side particulate fouling of heat exchangers. *Fouling of Heat Exchanger Surfaces, ed. Bryers, RW*, p. 257–275, 1983. Citado na página 5.

BERGMAN, T. L. et al. *Fundamentos de Transferência de Calor e de Massa*. [S.l.]: LTC, 2008. Citado 4 vezes nas páginas iv, 14, 15 e 16.

BIYANTO, T. R. et al. Modelling and simulation of industrial heat exchanger etworks under fouling condition using integrated neural network and hysys. *Jurnal Ilmiah Kursor*, v. 8, n. 1, 2015. Citado na página 8.

BOTT, T. R. *Fouling of heat exchangers*. [S.l.]: Elsevier, 1995. Citado 5 vezes nas páginas iv, 4, 19, 20 e 27.

BOTT, T. R.; GUDMUNDSSON, J. S. Rippled silica deposits in heat exchanger tubes. In: BEGEL HOUSE INC. *International Heat Transfer Conference Digital Library*. [S.l.], 1978. Citado na página 5.

BRASIL. Lei federal 13.589. 2018. Disponível em: <https://www2.camara.leg.br/legin/fed/lei/2018/lei-13589-4-janeiro-2018-786057-publicacaooriginal-154702-pl.html>. Citado na página 13.

CAMPBELL, G. S.; NORMAN, J. *An introduction to environmental biophysics*. [S.l.]: Springer Science & Business Media, 2012. Citado 2 vezes nas páginas iv e 22.

CANVA. 2020. Disponível em: <canva.com>. Citado 2 vezes nas páginas iv e 2.

CARRIER. Aquaedge 23xrv. 2020. Disponível em: <https://carrierdobrasil.com.br/produtos/aquaedge-23xrv>. Citado 2 vezes nas páginas iv e 10.

CHENOWETH, J. M. Liquid fouling monitoring equipment. In: *Fouling science and technology*. [S.l.]: Springer, 1988. p. 49–65. Citado na página 4.

CHO, K. et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. Citado na página 7.

CHUNG, J. et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. Citado na página 7.

CREMASCHI, L.; WU, X. Effect of fouling on the thermal performance of condensers and on the water consumption in cooling tower systems. *Heat Transfer Engineering*, Taylor & Francis, v. 36, n. 7-8, p. 663–675, 2015. Citado 3 vezes nas páginas 21, 30 e 46.

FENG, J. et al. Reconstruction of porous media from extremely limited information using conditional generative adversarial networks. *Physical Review E*, v. 100, 09 2019. Citado 2 vezes nas páginas iv e 24.

FIRDAUS, N.; PRASETYO, B. T.; LUCIANA, T. Chiller: Performance deterioration and maintenance. *Energy Engineering*, Taylor & Francis, v. 113, n. 4, p. 55–80, 2016. Citado na página 19.

FRYER, P.; SLATER, N. A novel fouling monitor. *Chemical Engineering Communications*, Taylor & Francis, v. 57, n. 1-6, p. 139–152, 1987. Citado na página 5.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning (Adaptive Computation and Machine Learning series)*. [S.l.]: e MIT Press, Cambridge, England, 2016. Citado na página 45.

GREFF, K. et al. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, IEEE, v. 28, n. 10, p. 2222–2232, 2016. Citado na página 7.

GREGORY, N.; SANFORD, K. *Heat Transfer*. [S.l.]: Cambridge University Press, 2008. Citado 3 vezes nas páginas 6, 14 e 15.

HARTY, D.; BOTT, T. Deposition and growth of microorganisms on simulated heat exchanger surfaces. *Fouling of Heat Transfer Equipment*, Hemisphere Pub, p. 334–344, 1981. Citado na página 5.

HEBB, D. O. The organization of behavior; a neuropsycholocigal theory. *A Wiley Book in Clinical Psychology*, v. 62, p. 78, 1949. Citado na página 7.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado 2 vezes nas páginas 7 e 26.

HURWITZ, J.; KIRSCH, D. *Machine Learning*. [S.l.]: John Wiley Sons, Inc., 2018. Citado na página 22.

IBRAHIM, H. Fouling in heat exchangers. *Matlab–a fundamental tool for scientific computing and engineering applications*, INTECH Open Access Publisher, v. 3, p. 57–96, 2012. Citado na página 8.

INTERNATIONAL, W.-C. T. Chiller efficiency: Condenser fouling. 2010. Disponível em: <https://water-cti.com/pdf/ESPG/CP4-Chiller-Fouling-Chart.pdf>. Citado 2 vezes nas páginas iv e 3.

ISOPPO, D. F.; BORGES, T. P.; MANTELLI, M. B. Development of a detailed thermal model for designing heat pipe heat exchangers. *Proceedings of the ECOS*, 2009. Citado na página 6.

JENKINS, A. *The measurement of high temperature condensation of salt*. Tese (Doutorado) — University of Birmingham, 1994. Citado na página 6.

JOFFILY, L. d. A. L. Caracterização do desempenho de compressores de refrigeração segundo um ciclo superaquecido de teste. 2007. Citado na página 6.

JOZEFOWICZ, R.; ZAREMBA, W.; SUTSKEVER, I. An empirical exploration of recurrent network architectures. In: PMLR. *International conference on machine learning*. [S.l.], 2015. p. 2342–2350. Citado na página 7.

KAZI, S. Fouling and fouling mitigation on heat exchanger surfaces. In: *Heat Exchangers-Basics Design Applications*. [S.l.]: IntechOpen, 2012. Citado 4 vezes nas páginas iv, 6, 18 e 19.

KERN, D.; SEATON, R. A theoretical analysis of thermal surface fouling. *British Chemical Engineering*, v. 4, n. 5, 1959. Citado 2 vezes nas páginas 6 e 20.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 31.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 7.

MELO, L.; PINHEIRO, J. Fouling tests: equipment and methods. *Fouling in heat exchange equipment*, American Society for Mechanical Engineering New York, p. 43–49, 1984. Citado na página 5.

MELONI, B. D. B.; SOUSA, L. G. L. d. Proposta de metodologia preditiva para otimização da limpeza de trocadores de calor resfriado a água. 2019. Citado 4 vezes nas páginas 6, 21, 27 e 46.

MOTT, I.; BOTT, T. R. The adhesion of biofilms to selected materials of construction for heat exchangers. In: *Proceedings 9th international heat transfer conference, Jerusalem*. [S.l.: s.n.], 1991. v. 5, p. 21–26. Citado na página 5.

NETTO, J. M. d. A. t. *Manual de Hidráulica*. [S.l.]: INTECH Open Access Publisher, 2011. Citado na página 1.

PATEL, T. *Examination of the changes in microbial distribution with time in cooling water slimes*. Tese (Doutorado) — University of Birmingham, 1981. Citado na página 4.

PEREIRA, G. d. S. Anáise numérica e experimental de sistema de ar condicionado em edifícios verdes. Universidade Federal de Pernambuco, 2016. Citado na página 6.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 7.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986. Citado na página 7.

SAHOO, D. et al. Online deep learning: Learning deep neural networks on the fly. *arXiv preprint arXiv:1711.03705*, 2017. Citado na página 46.

SHEN, C. et al. Fouling of enhanced tubes for condensers used in cooling tower systems: a literature review. *Applied Thermal Engineering*, Elsevier, 2015. Citado na página 19.

STOECKER, W. F. *Refrigeration and Air Conditioning*. [S.l.]: McGraw-Hill, 1983. v. 2. Citado 3 vezes nas páginas iv, 6 e 12.

SUNDAR, S. et al. Fouling modeling and prediction approach for heat exchangers using deep learning. *International Journal of Heat and Mass Transfer*, Elsevier, v. 159, p. 120112, 2020. Citado 3 vezes nas páginas iv, 8 e 9.

TOMPSON, J. et al. Accelerating eulerian fluid simulation with convolutional networks. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2017. p. 3424–3433. Citado na página 7.

TWOMINUTEPAPERS. How well can an ai learn physics? 2020. Disponível em: <https://www.youtube.com/watch?v=2Bw5f4vYL98&ab_channel=TwoMinutePap>. Citado 2 vezes nas páginas iv e 8.

VARSAMOPOULOS, S. et al. Designing neural network based decoders for surface codes. *arXiv preprint arXiv:1811.12456*, 2018. Citado 2 vezes nas páginas iv e 26.

WATKINSON, A.; MARTINEZ, O. Scaling of heat exchanger tubes by calcium carbonate. *Journal of Heat Transfer*, 1975. Citado na página 6.

Apêndices

A Código EES

codigo_ees

Equations

```
$UnitSystem SI Mass kJ C kPa
```

$c_a = 4,2 \ [\text{kJ/kg} \cdot \text{C}]$ calor específico

$\rho_a = 998 \ [\text{kg/m}^3]$ massa específica

Dados físicos do trocador

$N_t = 500$ número de tubos

$D_i = 0,015 \ [\text{m}]$ diâmetro interno

$D_o = 0,017 \ [\text{m}]$ diâmetro externo

$A_i = L \cdot pi\# \cdot D_i \cdot N_t$ área interna

$A_o = L \cdot pi\# \cdot D_o \cdot N_t$ área externa

$L = 5 \ [\text{m}]$ comprimento do tubo

$k_{cu} = 0,401 \ [\text{kW/m} \cdot \text{K}]$ condutividade térmica do cobre

$RelRough_{cu} = 0,005$

Dados de operação

$Q_{ev} = 0,0378 \ [\text{m}^3/\text{s}]$ vazão de água no evaporador

$\dot{m}_{a;ev} = \rho_a \cdot Q_{ev}$ vazão mássica de água no evaporador

$Q_{cd} = 0,031 \ [\text{m}^3/\text{s}]$ vazão de água no condensador

$\dot{m}_{a;cd} = \rho_a \cdot Q_{cd}$ vazão mássica de água no condensador

$$u = \frac{Q_{cd}/N_t}{\left(pi\# \cdot \left(\frac{(D_i)^2}{4}\right)\right)}$$ velocidade média

$P_{bomba} = 300 \ [\text{kPa}]$

$T_{o;cd} = 29,5 \ [\text{C}]$

$T_{o;ev} = 7 \ [\text{C}]$

$F\$ =$ 'R134A' fluido refrigerante

$\eta_c = 0,65$ eficiência do compressor

$\dot{m}_{ref} = 1,5$ vazão mássica de refrigerante

$UA_{ev} = 400$

$\dot{Q}_{ev} = 450$

Parâmetros de incrustação

$d = \left(7,9 \cdot 10^{-10}\right)\ \left[\mathrm{m}^2/\mathrm{s}\right]$

$k_{sp} = \left(10^{-9}\right)\ \left[\mathrm{mol}^2/\mathrm{L}^2\right]$ solubilidade

$dureza = 350$

$Ca_2 = (0,4004 \cdot dureza)$

$C_{o;3;2} = (0,5996 \cdot dureza)$

$k_f = 2,19\ \left[\mathrm{kW/m \cdot K}\right]$ condutividade térmica

$\rho_f = 2711\ \left[\mathrm{kg/m}^3\right]$ densidade do $CaCO_3$

$R_g = 1,986$ constante universal dos gases

$\mu_{agua} = 0,0008141\ \left[\mathrm{N \cdot s/m}^2\right]$

$TBU = \text{Lookup}(\text{‘dados’}\ ;\ TableRun\#;\ 1)$

Evaporador

$\dot{Q}_{ev} = \epsilon_{ev} \cdot \dot{m}_{a;ev} \cdot c_a \cdot (T_{i;ev} - T_4)$

$\dot{Q}_{ev} = \dot{m}_{a;ev} \cdot c_a \cdot (T_{i;ev} - T_{o;ev})$

$\epsilon_{ev} = 1 - \exp\left(-\dfrac{UA_{ev}}{(\dot{m}_{a;ev} \cdot c_a)}\right)$

Saída do evaporador - refrigerante como vapor saturado no caso incrustado

$T_1 = T_4$

$P_1 = \mathrm{P_{sat}}\left(F\$;\ \mathrm{T} = T_1\right)$

$x_1 = 1$

$s_1 = \mathrm{s}\left(F\$;\ \mathrm{T} = T_1;\ \mathrm{x} = x_1\right)$

$h_1 = \mathrm{h}\left(F\$;\ \mathrm{T} = T_1;\ \mathrm{x} = x_1\right)$

$v_1 = \mathrm{v}\left(F\$;\ \mathrm{T} = T_1;\ \mathrm{x} = x_1\right)$

Saída do compressor - refrigerante superaquecido no caso incrustado

$P_2 = P_3$

$s_{2s} = s_1$

$h_{2s} = \mathrm{h}\left(F\$;\ \mathrm{P} = P_2;\ \mathrm{s} = s_{2s}\right)$

$h_2 = h_1 + \dfrac{h_{2s} - h_1}{\eta_c}$

$$T_2 = \mathrm{T}\left(F\$;\ \mathrm{h} = h_2;\ \mathrm{P} = P_2\right)$$

$$s_2 = \mathrm{s}\left(F\$;\ \mathrm{h} = h_2;\ \mathrm{P} = P_2\right)$$

$$v_2 = \mathrm{v}\left(F\$;\ \mathrm{h} = h_2;\ \mathrm{P} = P_2\right)$$

Saída do condensador - refrigerante como líquido saturado no caso incrustado

$$P_3 = \mathrm{P_{sat}}\left(F\$;\ \mathrm{T} = T_3\right)$$

$$x_3 = 0$$

$$s_3 = \mathrm{s}\left(F\$;\ \mathrm{T} = T_3;\ \mathrm{x} = x_3\right)$$

$$h_3 = \mathrm{h}\left(F\$;\ \mathrm{T} = T_3;\ \mathrm{x} = x_3\right)$$

$$v_3 = \mathrm{v}\left(F\$;\ \mathrm{T} = T_3;\ \mathrm{x} = x_3\right)$$

Saída da válvula de expansão no caso incrustado

$$h_4 = h_3$$

$$P_4 = P_1$$

$$s_4 = \mathrm{s}\left(F\$;\ \mathrm{h} = h_4;\ \mathrm{P} = P_4\right)$$

$$v_4 = \mathrm{v}\left(F\$;\ \mathrm{h} = h_4;\ \mathrm{P} = P_4\right)$$

Calor rejeitado no caso incrustado

$$\dot{Q}_{cd} = \dot{Q}_{ev} + \dot{W}$$

COP no caso incrustado

$$\dot{W} = \dot{Q}_{ev}/COP + 0{,}000000001$$

Condensador no caso incrustado

$$\dot{Q}_{cd} = \epsilon_{cd} \cdot \dot{m}_{a;cd} \cdot c_a \cdot \left(T_3 - TBU\right)$$

$$\dot{Q}_{cd} = \dot{m}_{a;cd} \cdot c_a \cdot \left(T_{o;cd} - TBU\right)$$

$$\epsilon_{cd} = 1 - \exp\left(-\frac{UA_t}{\left(\dot{m}_{a;cd} \cdot c_a\right)}\right)$$

$$UA_t = \left(\frac{1000}{\left(H_{i;t} \cdot A_{i;t}\right)} + \frac{\ln\left(D_o/D_{i;t}\right)}{\left(2 \cdot pi\# \cdot k_{cu} \cdot L \cdot N_t\right)} + R_f/A_{i;t} + \frac{1000}{H_o \cdot A_o}\right)^{-1}$$

Progressão

$$\dot{m}_d = \left(k_d \cdot C_{o;3;2} \cdot \frac{1 - \frac{k_{sp}}{Ca_2 \cdot C_{o;3;2}}}{1 + \left(\frac{k_d}{k_r \cdot C_{o;3;2}}\right) + \frac{C_{o;3;2}}{\left(Ca_2\right)}}\right)$$

$$T_{med} = \left(\frac{TBU + T_{o;cd}}{2}\right) + 273$$

$$k_r = \exp\left(38,74 - \left(\frac{20700}{R_g \cdot T_{med}}\right)\right)$$

$$k_d = 0,023 \cdot u_t \cdot Re_t^{-0,17} \cdot Sc^{-0,67}$$

$$Sc = \frac{u_t \cdot D_i}{(Re_t \cdot d)}$$ número de Schmidt

$$Re = \rho_a \cdot u \cdot D_i / \mu_{agua}$$ número de Reynolds

$$c_r = \frac{0,00212 \cdot \left((u_t)^2\right)}{k_f^{0,5}}$$

$$dRdt = \left(\frac{\dot{m}_d - c_r \cdot \rho_f \cdot k_f \cdot R_f}{\rho_f \cdot k_f}\right)$$

$$R_f = 0 + \int_0^{tablerun\#} dRdt \; \mathrm{d}t$$

$$\frac{\Delta P_f}{\Delta P_c} = \left(\frac{(1,1 \cdot f_c)}{f_c}\right) \cdot (D_i / D_{i;t})^5$$

$D_{i;t} = D_i - (2 \cdot x_t)$ diâmetro interno

$x_t = R_f \cdot k_f$

$A_{i;t} = L \cdot pi\# \cdot D_{i;t} \cdot N_t$

$$u_t = \frac{Q_{cd}/N_t}{\left(pi\# \cdot \left(\frac{(D_{i;t})^2}{4}\right)\right)}$$ velocidade média

call $PipeFlow($'water' ; $TBU;\ P_{bomba};\ (\dot{m}_{a;cd}/N_t)\,;\ D_i;\ L;\ RelRough_c)\ u$

$H_{i;t};\ h_{H;t};\ \Delta P_t;\ Nusselt_T;\ f_t;\ Re_t)$

call $Cond_{horizontal;N;Cylinders}($'R134a' ; $T_3;\ T_3 - 0,01;\ D_o;\ )\ 1$

$H_o;\ Nusselt_m)$

Saída do evaporador - refrigerante como vapor saturado no caso limpo

$T_{1;c} = T_4$

$P_{1;c} = \mathrm{P_{sat}}\,(F\$;\ \mathrm{T} = T_{1;c})$

$x_{1;c} = 1$

$s_{1;c} = \mathrm{s}\,(F\$;\ \mathrm{T} = T_{1;c};\ \mathrm{x} = x_{1;c})$

$h_{1;c} = \mathrm{h}\,(F\$;\ \mathrm{T} = T_{1;c};\ \mathrm{x} = x_{1;c})$

$v_{1;c} = \mathrm{v}\,(F\$;\ \mathrm{T} = T_{1;c};\ \mathrm{x} = x_{1;c})$

Saída do compressor - refrigerante superaquecido no caso limpo

$P_{2;c} = P_{3;c}$

$s_{2s;c} = s_{1;c}$

$h_{2s;c} = \mathrm{h}\left(F\$;\ \mathrm{P} = P_{2;c};\ \mathrm{s} = s_{2s;c}\right)$

$h_{2;c} = h_{1;c} + \dfrac{h_{2s;c} - h_{1;c}}{\eta_c}$

$T_{2;c} = \mathrm{T}\left(F\$;\ \mathrm{h} = h_{2;c};\ \mathrm{P} = P_{2;c}\right)$

$s_{2;c} = \mathrm{s}\left(F\$;\ \mathrm{h} = h_{2;c};\ \mathrm{P} = P_{2;c}\right)$

$v_{2;c} = \mathrm{v}\left(F\$;\ \mathrm{h} = h_{2;c};\ \mathrm{P} = P_{2;c}\right)$

Saída do condensador - refrigerante como líquido saturado no caso limpo

$P_{3;c} = \mathrm{P_{sat}}\left(F\$;\ \mathrm{T} = T_{3;c}\right)$

$x_{3;c} = 0$

$s_{3;c} = \mathrm{s}\left(F\$;\ \mathrm{T} = T_{3;c};\ \mathrm{x} = x_{3;c}\right)$

$h_{3;c} = \mathrm{h}\left(F\$;\ \mathrm{T} = T_{3;c};\ \mathrm{x} = x_{3;c}\right)$

$v_{3;c} = \mathrm{v}\left(F\$;\ \mathrm{T} = T_{3;c};\ \mathrm{x} = x_{3;c}\right)$

Saída da válvula de expansão no caso limpo

$h_{4;c} = h_{3;c}$

$P_{4;c} = P_{1;c}$

$s_{4;c} = \mathrm{s}\left(F\$;\ \mathrm{h} = h_{4;c};\ \mathrm{P} = P_{4;c}\right)$

$v_{4;c} = \mathrm{v}\left(F\$;\ \mathrm{h} = h_{4;c};\ \mathrm{P} = P_{4;c}\right)$

Calor rejeitado no caso limpo

$\dot{Q}_{cd;c} = \dot{Q}_{ev} + \dot{W}_c$

COP no caso limpo

$\dot{W}_c = \dot{Q}_{ev}/COP_c + 0,000000001$

Condensador no caso limpo

$\dot{Q}_{cd;c} = \epsilon_{cd;c} \cdot \dot{m}_{a;cd} \cdot c_a \cdot \left(T_{3;c} - TBU\right)$

$\dot{Q}_{cd;c} = \dot{m}_{a;cd} \cdot c_a \cdot \left(T_{o;cd} - TBU\right)$

$\epsilon_{cd;c} = 1 - \exp\left(-\dfrac{UA_c}{\left(\dot{m}_{a;cd} \cdot c_a\right)}\right)$

$UA_c = \left(\dfrac{1000}{\left(H_{i;c} \cdot A_i\right)} + \dfrac{\ln\left(D_o/D_i\right)}{\left(2 \cdot pi\# \cdot k_{cu} \cdot L \cdot N_t\right)} + \dfrac{1000}{H_{o;c} \cdot A_o}\right)^{-1}$

call $PipeFlow$(‘water’ ; TBU; P_{bomba}; $\left(\dot{m}_{a;cd}/N_t\right)$; D_i; L; $RelRough_c$) u

$H_{i;c}$; $h_{H;c}$; ΔP_c; $Nusselt_c$; f_c; Re_c)

call $Cond_{horizontal;N;Cylinders}$(‘R134a’ ; $T_{3;c}$; $T_{3;c} - 0,01$; D_o;) 1

$H_{o;c}$; $Nusselt_{m;c}$)

B Código Rede Neural

Bloco 1 - bibliotecas

```
%load_ext tensorboard
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import pandas as pd
import os

from sklearn.preprocessing import MinMaxScaler

from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Dense, Bidirectional, Embedding, LSTM
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard, ReduceLROnPlateau
from tensorflow.keras.backend import square, mean
import tensorflow_datasets as tfds
from keras.callbacks import TensorBoard
from tensorboard.plugins.hparams import api as hp

import time
import seaborn as sns
import datetime
import io

from google.colab import files
```

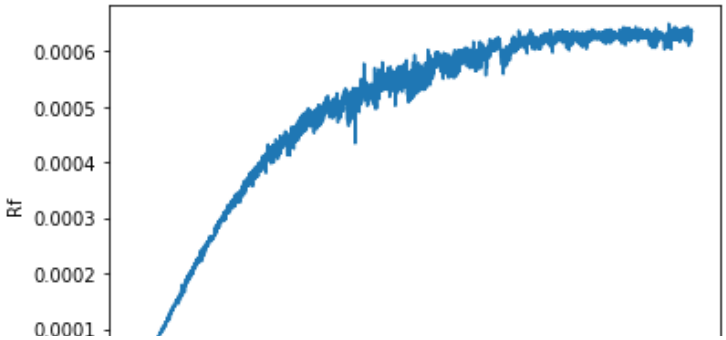
Bloco 2 - importação dos dados

```
uploaded = files.upload()

#dataframe
df = pd.read_csv(
    io.BytesIO(uploaded['dados_incrustacao1.csv']),
    #names=["timestamp","TBU","Rf","Numero","Comprimento","Vazao","Condensacao"])
    parse_dates=['timestamp'],
    index_col="timestamp"
    )
df.shape
print(df.head())
```

```
Escolher arquivos Nenhum arquivo selecionado Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving dados_incrustacao1.csv to dados_incrustacao1 (1).csv
                     Rf     UA  Tcond    TBU    P  L
timestamp
1          2.200000e-07  181.6  32.85  19.31  300  5
2          4.390000e-07  181.6  32.85  19.31  300  5
3          6.550000e-07  179.8  33.09  18.78  300  5
4          8.680000e-07  178.2  33.31  18.31  300  5
```

Bloco 3 - Rf em função do tempo

```
sns.lineplot(x=df.index,y='Rf',data=df);
```



Bloco 4 - Separação de dados de treino e teste

0 2000 4000 6000 8000 10000

```
#porcentagem de treinos e testes
train_size = int(len(df)*0.9)
test_size = len(df) - train_size

#separa as variáveis de treinos e testes
train, test = df.iloc[0:train_size], df.iloc[train_size:len(df)]

print(len(train), len(test))
```

```
9000 1000
```

Bloco 5 - Pré processamento dos dados

```
#This Scaler removes the median and scales the data according to the quantile range (defaults to IQR: Inter
#Reference: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html
from sklearn.preprocessing import RobustScaler

#especifica as colunas usadas com o Scaler
f_columns = ['UA', 'Tcond', 'TBU', 'P', 'L']

#feature transformer - chama o method fit nos dados de treino (somente) e o transforma
f_transformer = RobustScaler()
Rf_transformer = RobustScaler()

#transforma a string de dados em um array de numpy
f_transformer = f_transformer.fit(train[f_columns].to_numpy())
Rf_transformer = Rf_transformer.fit(train[['Rf']])

#transforma os dados de treino e output
train.loc[:,f_columns] = f_transformer.transform(train[f_columns].to_numpy())
#retorna Rf de treino normalizado e armazena na coluna Rf do train
train['Rf'] = Rf_transformer.transform(train[['Rf']])

#transforma os dados de teste
test.loc[:,f_columns] = f_transformer.transform(test[f_columns].to_numpy())
#retorna Rf de teste normalizado e armazena na coluna Rf do test
test['Rf'] = Rf_transformer.transform(test[['Rf']])

```

Bloco 6 - função de que divide os dados em intervalos

```
#função para cortar a série temporal em séries menores
def create_dataset(X, y, time_steps=1):

  Xs, ys = [], []

#Loop que percorrerá cada dado em cada subsequência
  for i in range(len(X)-time_steps): #percorre o tamanho de train menos o número de time_steps
```

```
    dado_dia = X.iloc[i: (i+time_steps)].to_numpy() #dado_dia conterá as 24 horas correspondentes do loop p
    # i=1, dado_dia = 1:25
    # i=2, dado_dia = 2:26
    # i=3, dado_dia = 3:27
    Xs.append(dado_dia)
    #Xs[0] = 0:24
    #Xs[1] = 1:25
    #Xs[2] = 2:26
    ys.append(y.iloc[i + time_steps])
    #ys[0] = 0+24 = 24
    #ys[1] = 25
    #ys[2] = 26
  return np.array(Xs), np.array(ys)
```

Bloco 7 - definição final dos dados de treino e teste

```
TIME_STEPS = 24

X_train, y_train = create_dataset(train, train.Rf, time_steps=TIME_STEPS)
X_test, y_test = create_dataset(test, test.Rf, time_steps=TIME_STEPS)
```

Bloco 8 - prints para verificações

```
# [samples, times_steps, n_features]

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(8976, 24, 6) (8976,)
(976, 24, 6) (976,)
```

Bloco 9 - Arquitetura da rede neural

```
def train_model(hparams):
  units = hparams[HP_NUM_UNITS]
  learning_rate = hparams[HP_LR]

  model = keras.Sequential()

  model.add(
          keras.layers.Bidirectional(
              keras.layers.LSTM(
                  units=units,
                  input_shape=(X_train.shape[1],X_train.shape[2]))))

  model.add(keras.layers.Dropout(rate=0.2))

  model.add(keras.layers.Dense(units=1))

  #função de custo e otimizador
  model.compile(loss='mean_squared_error',
                    optimizer=keras.optimizers.Adam(lr=learning_rate))

  NAME = str(units) + "units_" + str(learning_rate) + "learning_rate"
  print(NAME)

  #caminho e nome da pasta em que deve ser salvo
  path = "logs/fit1/" + NAME

  #Hparams
  with tf.summary.create_file_writer(path).as_default():
```

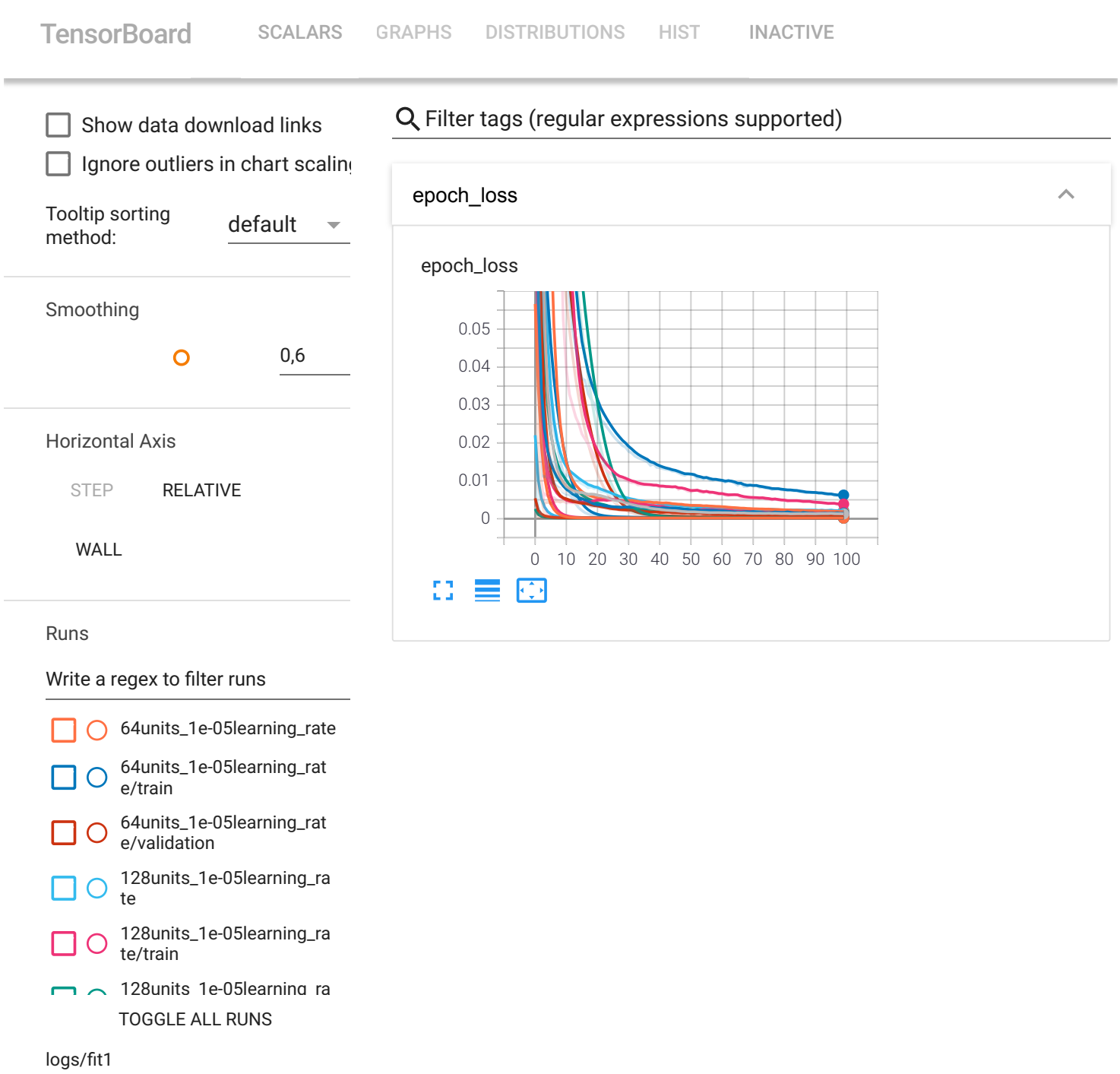
```
  with tf.summary.create_file_writer(path).as_default():
    hp.hparams(hparams)

  #histograma para cada época
  tensorboard = tf.keras.callbacks.TensorBoard(log_dir = path, histogram_freq=1)

  history = model.fit(
    X_train, y_train,
    batch_size=64,
    epochs=100,
    validation_split=0.1,
    shuffle=False,
    callbacks=[tensorboard])

HP_NUM_UNITS = hp.HParam("num units", hp.Discrete([64, 128, 256]))
HP_LR = hp.HParam("learning rate", hp.Discrete([1e-3, 1e-4, 1e-5]))

for lr in HP_LR.domain.values:
  for units in HP_NUM_UNITS.domain.values:
      hparams = {
          HP_LR : lr,
          HP_NUM_UNITS : units
      }

      train_model(hparams)
```

```
64units_1e-05learning_rate
Epoch 1/100
127/127 [==============================] - 7s 23ms/step - loss: 1.4433 - val_loss: 0.1459
Epoch 2/100
127/127 [==============================] - 1s 7ms/step - loss: 1.2914 - val_loss: 0.1385
Epoch 3/100
127/127 [==============================] - 1s 7ms/step - loss: 1.1083 - val_loss: 0.1301
Epoch 4/100
127/127 [==============================] - 1s 7ms/step - loss: 0.9385 - val_loss: 0.1213
Epoch 5/100
127/127 [==============================] - 1s 7ms/step - loss: 0.7691 - val_loss: 0.1122
Epoch 6/100
127/127 [==============================] - 1s 7ms/step - loss: 0.6200 - val_loss: 0.1029
Epoch 7/100
127/127 [==============================] - 1s 7ms/step - loss: 0.4975 - val_loss: 0.0935
Epoch 8/100
127/127 [==============================] - 1s 7ms/step - loss: 0.3728 - val_loss: 0.0841
Epoch 9/100
127/127 [==============================] - 1s 7ms/step - loss: 0.2961 - val_loss: 0.0748
Epoch 10/100
127/127 [==============================] - 1s 7ms/step - loss: 0.2249 - val_loss: 0.0659
Epoch 11/100
127/127 [==============================] - 1s 7ms/step - loss: 0.1697 - val_loss: 0.0576
Epoch 12/100
127/127 [==============================] - 1s 7ms/step - loss: 0.1372 - val_loss: 0.0499
Epoch 13/100
127/127 [==============================] - 1s 7ms/step - loss: 0.1118 - val_loss: 0.0430
Epoch 14/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0967 - val_loss: 0.0369
Epoch 15/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0835 - val_loss: 0.0318
Epoch 16/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0743 - val_loss: 0.0273
Epoch 17/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0745 - val_loss: 0.0233
Epoch 18/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0660 - val_loss: 0.0198
Epoch 19/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0628 - val_loss: 0.0168
Epoch 20/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0598 - val_loss: 0.0143
Epoch 21/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0550 - val_loss: 0.0120
```

```
Epoch 22/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0512 - val_loss: 0.0100
Epoch 23/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0502 - val_loss: 0.0083
Epoch 24/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0470 - val_loss: 0.0069
Epoch 25/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0449 - val_loss: 0.0057
Epoch 26/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0440 - val_loss: 0.0047
Epoch 27/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0424 - val_loss: 0.0039
Epoch 28/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0392 - val_loss: 0.0032
Epoch 29/100
127/127 [==============================] - 1s 7ms/step - loss: 0.0377 - val_loss: 0.0027
```

Bloco 10 - tensorboard

```
1 %tensorboard --logdir logs/fit1
```



Bloco 11 comparação entre valores de teste reais e calculados (para um dos testes)

```
model2 = keras.Sequential()

model2.add(
        keras.layers.Bidirectional(
            keras.layers.LSTM(
                units=128,
                input_shape=(X_train.shape[1],X_train.shape[2]))))

model2.add(keras.layers.Dropout(rate=0.2))

model2.add(keras.layers.Dense(units=1))

#função de custo e otimizador
model2.compile(loss='mean_squared_error',
                  optimizer=keras.optimizers.Adam(lr=1e-4))

history = model2.fit(
  X_train, y_train,
  batch_size=64,
  epochs=30,
  validation_split=0.1,
  shuffle=False)

y_pred = model2.predict(X_test)


#inverter o scalar
y_train_inv = Rf_transformer.inverse_transform(y_train.reshape(1, -1))
y_test_inv = Rf_transformer.inverse_transform(y_test.reshape(1, -1))
y_pred_inv = Rf_transformer.inverse_transform(y_pred)

#plot y_pred
plt.plot(y_test_inv.flatten(), marker='.', label='true')
plt.plot(y_pred_inv.flatten(), 'r', label='predicted')
plt.legend();


```

```
Epoch 1/30
127/127 [==============================] - 5s 14ms/step - loss: 1.4060 - val_loss: 0.0793
Epoch 2/30
127/127 [==============================] - 1s 7ms/step - loss: 0.1090 - val_loss: 0.0330
Epoch 3/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0435 - val_loss: 0.0117
Epoch 4/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0296 - val_loss: 0.0035
Epoch 5/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0212 - val_loss: 0.0013
Epoch 6/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0177 - val_loss: 6.7305e-04
Epoch 7/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0157 - val_loss: 4.8361e-04
Epoch 8/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0165 - val_loss: 4.7283e-04
Epoch 9/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0161 - val_loss: 3.7908e-04
Epoch 10/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0155 - val_loss: 3.4618e-04
Epoch 11/30
127/127 [==============================] - 1s 8ms/step - loss: 0.0177 - val_loss: 3.1555e-04
Epoch 12/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0175 - val_loss: 3.0325e-04
Epoch 13/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0191 - val_loss: 2.9667e-04
Epoch 14/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0175 - val_loss: 2.7399e-04
Epoch 15/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0182 - val_loss: 2.6670e-04
Epoch 16/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0173 - val_loss: 2.4721e-04
Epoch 17/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0170 - val_loss: 2.4731e-04
Epoch 18/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0167 - val_loss: 2.3903e-04
Epoch 19/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0169 - val_loss: 2.3464e-04
Epoch 20/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0156 - val_loss: 2.2609e-04
Epoch 21/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0141 - val_loss: 2.2010e-04
Epoch 22/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0128 - val_loss: 2.1245e-04
Epoch 23/30
127/127 [==============================] - 1s 7ms/step - loss: 0.0137 - val_loss: 2.2445e-04
```

Bloco 12 - visualização geral dos dados (para um dos testes)

```
Epoch 25/30
```

```
plt.plot(np.arange(0, len(y_train)), y_train_inv.flatten(), 'g', label="history")
plt.plot(np.arange(len(y_train), len(y_train) + len(y_test)), y_test_inv.flatten(), marker='.', label="true
plt.plot(np.arange(len(y_train), len(y_train) + len(y_test)), y_pred_inv.flatten(), 'r', label="prediction"
plt.legend();
```